1988-12

Microcomputer applications with PC LAN in battleships

Gulesen, Nevzat

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

MICROCOMPUTER APPLICATIONS
WITH
PC LANS
IN BATTLESHIPS

by

NEVZAT GULESEN

December 1988

Thesis Advisor: Prof. U. Kodres

Approved for public release; distribution unlimited

:lassified

rity classification of this page

REPORT DOCUMENTATION PAGE

Report Security Classification Unclassified		1b Restrictive Markings	
Security Classification Authority		3 Distribution Availability of Report	
Declassification Downgrading Schedule		Approved for public release; distribution is unlimited.	
erforming Organization Report Number(s)		5 Monitoring Organization Report Number(s)	
Name of Performing Organization ıval Postgraduate School	6b Office Symbol *(if applicable)* 33	7a Name of Monitoring Organization Naval Postgraduate School	
Address *(city, state, and ZIP code)* onterey, CA 93943-5000		7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	
Name of Funding Sponsoring Organization	8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number	
Address *(city, state, and ZIP code)*		10 Source of Funding Numbers	
		Program Element No	Project No / Task No / Work Unit Accession No

Title *(include security classification)* MICROCOMPUTER APPLICATIONS WITH PC LAN IN BATTLESHIPS

Personal Author(s) Nevzat Gulesen

Type of Report	13b Time Covered	14 Date of Report *(year, month, day)*	15 Page Count
aster's Thesis	From To	December 1988	164

Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or po-
on of the Department of Defense or the U.S. Government.

Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
ld	Group	Subgroup	LAN, CIC, Damage Control

Abstract *(continue on reverse if necessary and identify by block number)*

This thesis explores the hardware requirements of a local area network and then constructs a multiuser software library
ckage for Turkish Battleships. The software implementation is designed as an expandable package so that future require-
nts can be met. The software package consists of three major parts. These are Personnel Evaluations, Combat Information
nter and Damage Control. Listings of the programs developed are presented, as well as instructions for their effective use.
It is concluded that a PC Local Area Network with the proper library programs is feasible for Turkish Battle Ships'
mputing requirements.

Distribution Availability of Abstract unclassified unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
Name of Responsible Individual of. Uno Kodres	22b Telephone *(include Area code)* (408) 646-2197	22c Office Symbol 52Kr

FORM 1473,84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

security classification of this page

Unclassified

Approved for public release; distribution is unlimited.

Microcomputer Applications with PC LAN in Battleships

by

Nevzat Gulesen
Lieutenant J.G. Turkish Navy
B.S., Turkish Naval Academy, 1982

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1988

ABSTRACT

This thesis explores the hardware requirements of a local area network and then constructs a multiuser software library package for Turkish Battleships. The software implementation is designed as an expandable package so that future requirements can be met. The software package consists of three major parts. These are Personnel Evaluations, Combat Information Center and Damage Control. Listings of the programs developed are presented, as well as instructions for their effective use.

It is concluded that a PC Local Area Network with the proper library programs is feasible for Turkish Battle Ships' computing requirements.

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every of effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they can not be considered validated. Any application of these programs without additional verification is at the risk of the user.

Some terms used in this research are registered trademarks of commercial products. Rather than attempt to cite a trademark, all trademarks appearing in this thesis are listed below, following the firm holding the trademark :

1. International Business Machines Corporation
   IBM
   IBM PC/AT
2. Microsoft Corporation
   MS-DOS
3. Intel Corporation
   Intel iAPx 8086, 80186, 80286, 80386, 80387
4. Ashton Tate
   DataBase III +
5. Borland International
   Turbo Pascal 4.0
   Turbo Pascal 5.0
6. WordPerfect Corporation
   WordPerfect

TABLE OF CONTENTS

I. INTRODUCTION 1
A. BACKGROUND 1
B. DEFINITION OF THE PROBLEM 1
C. RECOMMENDED SOLUTION 2
D. STRUCTURE 3

II. INTRODUCTION TO LAN 4
A. DEFINITION 4
1. Peripheral and Information Sharing 4
2. Characteristics of PC Network 4
3. Major Advantages of the Local Area Network 5
B. EVALUATION OF LOCAL AREA NETWORKS 5
C. TRANSMISSION TECHNIQUES 6
D. MEDIUM ACCESS CONTROL METHODS 6
1. CSMA CD 6
2. Control Token 7
3. Slotted Ring 7
E. NETWORK CONFIGURATION 7
1. Memory and Disk Requirements 8
2. Operating System 8
3. Users 8
4. Transmission Medium 9
F. IMPLEMENTATION 9
G. INTEL 80386 MICROPROCESSOR ARCHITECTURE 9
1. Cache Strategy 10
2. Memory Management 10
3. Multitasking 11
4. Software Compatibility 11
5. Performance and Conclusion 11
H. LOCATION IN THE SHIP 11

III. THE SOFTWARE IMPLEMENTATION 14
A. DEFINITION OF THE PROBLEM 14
B. RECOMMENDED SOLUTION 14
C. GENERAL CHARACTERISTICS OF THE PACKAGE 14
1. Language and Software Compatibility 14
2. Hardware Compatibility 15
3. Math-coprocessor 16
4. Implementation 16
D. STATISTICAL ANALYSIS 17

IV. USER MANUAL 19
A. GETTING STARTED WITH TCG. MF. CAKMAK 19
B. COMBAT INFORMATION CENTER 20
C. PERSONNEL EVALUATIONS 26
D. DAMAGE CONTROL 28
1. General Ship Information 28
2. Draft Diagram and Functions. 28
3. Tank Diagrams 28
4. Fire Fighting Exercise 28
5. Liquid Loading Diagram 29
6. Exit 30
E. SOURCE CODE 31

V. CONCLUSIONS AND RECOMMENDATIONS 151
A. CONCLUSIONS 151
B. RECOMMENDATIONS AND FUTURE IMPLEMENTATIONS 152

REFERENCES 153

INITIAL DISTRIBUTION LIST 154

LIST OF FIGURES

Figure 1. PC Locations in the Battleship. 12
Figure 2. Menu Options in TCG. MF. Cakmak. 20
Figure 3. CIC Data Entry Form. 21
Figure 4. A Sample Radar Screen and Solution. 23
Figure 5. CIC Data Entry Table for Pivotship Solution. 24
Figure 6. Finite State Automata Model for CIC States. 25
Figure 7. Personnel Evaluations Data Entry Form. 27

ACKNOWLEDGEMENTS

Many people have contributed to the completion of this thesis. I would like to thank the following people:

Prof. Uno Kodres, Thesis Advisor, Computer Science Department.

Mr. Ben Mortagy, Adjunct Instructor, Administrative Science Department.

Lt.J.G Cengiz Sengel, Former CIC Officer, Turkish Navy.

Lt.J.G Sukru Korlu, Former CIC Officer, Turkish Navy.

Lt.J.G Mumtaz Tunc, Former EW Officer, Turkish Navy.

Capt. Rod Scott, Canadian Air Force.

Lt. Martin Buker, USN.

Mrs. Jackie Kite.

Mrs. Emine Gulesen.

I. INTRODUCTION

A. BACKGROUND

In the recent years, the use of the computers for military purposes has grown rapidly, because they can do the computations better, faster and more accurate than human beings. As the time scale of hypothetichal battle shrinks, it becomes more clear that the traditional modes of battle planning and management are no longer practical. Many of these problems, like accuracy, less response time, information exchanging and communication have been solved in military applications as the computer technology advances.

On the other hand, these advances in computer technology result in increases in cost. The budgets of many countries have been suffering from the overload of military expenses. The same problem has affected Turkey as well. Most of the Turkish destroyers were made in the United States during and right after the Second World War. In order to increase the effectivity and the functionality of these battleships, Turkey has been upgrading and modernizing the equipment located in these ships as well as buying and building new battleships. Introducing the digital computers to the battle ships is both a requirement and a part of these upgrading and modernization efforts.

B. DEFINITION OF THE PROBLEM

In increasing the effectivity and the functionality of Turkish battle ships, introducing digital computers will be one of the most important steps to take. Many of the ships duties can be easily handled by digital computers: faster, better, and more accurately. Introducing the digital computers will bring attention to three major considerations though:

1. A multiuser computing environment,
2. Military oriented proper application software libraries,
3. Cost effectivity.

The battle ships will require a multiuser computing environment, because ship duties are multifunctional. There will be many users working at a time. There is a number of multiuser computer systems available on the market, but most of them have huge hardware units which makes them hard to install into the battleships. The lack of space in the battleships, is a consideration. Most of the multiuser computing systems also require

special airconditioning. In case of damage of any hardware unit or airconditioning unit under war conditions, the entire system may not function very well.

One of the important aspects of the multiuser computer systems, is that they are too expensive to afford for each battle ship in the navy. Optimizing the cost of the computing requirements has been quite a big problem to solve.

Having a cost effective, very advanced type of multiuser computing system would not solve all the computing problems of Turkish Battleships.

The most important requirement is that of military oriented software, because the hardware units can be designed by the civilian companies, but military software can barely be implemented by civilian companies. The military oriented software generally includes confidential data and programs. This important restriction forces military associations to implement their own software requirement as well as using the civilian software. No matter how great, or how cost effective the multiuser systems we have in our battleships, the most important requirement would be the proper, military oriented application software library packages.

C. RECOMMENDED SOLUTION

A Local Area Network (LAN) consisting of a number of advanced type personal computers is one solution to both multiuser environment and a cost elimination problem. A LAN is a number of personal computers linked together, so that they can exchange information and share resources. In particular, they can share disk storage, data and peripheral devices like printers. The network permits data and programs to be transferred easily. A LAN with eight Intel 80386 microprocessor based computers will greatly increase the Turkish BattleShips' functions in speed and accuracy while adding the power of the digital computers.

In this thesis, we have implemented the most important three major application software packages for Turkish Battleships. These are:

1. Combat Information Center,
2. Personnel Evaluations,
3. Damage Control Center.

The lack of the military oriented application software problem, can be solved by continually implementing them. The computer users are not computer specialists, nor programmers on the battle ships. The implementation of the application software packages will increase the efficiency and the functionality of the battle ships.

D. STRUCTURE

Chapter 2 describes and explores the hardware requirements of a local area network. It also explains the major hardware characteristics of Intel 80386 microprocessor. Chapter 3 defines and discusses the military oriented software problem and recommends a solution. Evaluation and the statistical analysis of the software package created for Turkish battle ships are also presented in the Chapter 3. Chapter 4 provides the user manual and the source code. Recommendations, conclusions and future implementations are presented in Chapter 5.

II. INTRODUCTION TO LAN

This chapter will first introduce the LAN and its hardware requirements and then it will present the major features of Intel 80386 micropressor. At the end of the chapter, a network model for Turkish Battleships is presented.

A. DEFINITION

A Local Area Network is a number of personal computers linked together so that they can share resources and exchange information. A LAN is called local area network because the physical link between computers is limited. The users of the network can continue to work on their respective computers independently. This is the main distinction between a time-sharing computer system and a local area network. In a time-sharing system, various users of the system can not do their own work, every job processed is under the control of time-sharing system.

1. Peripheral and Information Sharing

Network computers can share resources such as disk drives, files, programs, printers and other peripherals. The devices are connected to the network indirectly. The main elements of the network are computers that are linked directly to the network. Peripheral and resource sharing is a very important concept in computer architecture and operating system analysis. Historically, Banker's Algorithm, Dining Philosophers algorithms have been provided to solve deadlock and other resource sharing problems. Effective sharing of the resources and the peripherals will reduce the total and the marginal cost of the system.

Even with two computers, a network has advantages. The maximum number of the computers that can be installed in a network depends on the so-called Network Adapter Card and the capacity of the network connection cable. Today, the maximum number of the computers in a local area network can be as many as 1000 within a radius of a few miles. One important limitation is that of the number of computers which are linked through the server. IBM recommends not to have more than 25 users connected to a server. The number of users connected to a server may increase in future.

2. Characteristics of PC Network

Each PC must have a Network Adapter Card in it.

Each PC must run the PC Network program to manage its participation in the network.

Each PC must have a cable to connect it to the network. The physical location of the computer in the network is not really important. Each computer is introduced to the network by its name [Ref. 1].

3. Major Advantages of the Local Area Network

1. Users can share the use of specialized equipment,
2. Programs and data can be stored as one master copy to be used by all the users of the network. There is no need for each user to have an individual copy of the commonly used programs. This saves a great amount of file space.
3. PCs get more power from sharing the resources. Individual PCs do not have to have fixed disks in them. Data and programs can be stored only in the servers' fixed disks. These programs, data and files can be retrieved as the network computers want to run them.
4. Data can be transported easily among the network computers. Network allows data to be transferred directly.

B. EVALUATION OF LOCAL AREA NETWORKS

The network should be able to run the programs properly. The network we are modeling for the Turkish Battleships, should be able to run the program constructed and presented in Chapter 4. The subject program as mentioned before, consists of three major parts. The network must be capable to run even the future software implementations. Any IBM oriented software has to be run in the network environment without causing any problem.

The network must be flexible to meet the changing and developing goals and the functions of the Turkish Battleships. It should have the capability of easy updating and modernizing features for the future advances in computer architectures and network configurations.

System should be simple to decrease the hardware, software and user problems. The users of the network are generally not computer professionals. Any hardware or software problem occurring at sea or in an emergency situation, must be solved by the ship personnel.

The elements of the network, must be able to communicate with other network elements. That will require compatibility. As stated before, Turkish Battle Ships will be employing strictly IBM products or their equivalents.

The above conditions can be reflected into the following parameters:

1. Data Transmission Rate is at least 1Mbps
2. Number of connected computers 7 + 1 Server
3. Low error rates 10^{-10}

4. Maximum node to node distance 120 meters. (Ship is about 119meters)
5. High system availability (over 95 %)
6. High System reliability
7. Intel 80386 microprocessors
8. Intel 80387 Math-coprocessors
9. Operating System DOS 3.30 to be upgraded later to allow the full power utilization of the microprocessor.

C. TRANSMISSION TECHNIQUES

There are two transmission techniques, baseband and broadband. Baseband employs bandwidths with no more than 50 MHz and bandwidth is taken up by transmitting signal. The bandwidth of the broadband is generally greater than baseband. It is generally the order of 300 Mhz and more. The aim of the broadband system is that it can provide a Local Area Network with the capability of handling large numbers of devices over up to 9-10 miles and carry image, data and voice transmissions. The network model in the Turkish Battleships will employ the broadband technique.

D. MEDIUM ACCESS CONTROL METHODS

The access method is one of the most important aspects of local area design. Only one device can successfully transmit on a shared medium at one time. Three tecniques are largely used in local area design.

1. CSMA/CD

Carrier Sense Multiple Access with Collision Detection is used with bus network topologies. A station can transmit if the line is free; otherwise it must wait. The waiting station should either back off for a specified time interval before listening again or keep monitoring until the line is clear to send. Because of propagation delay, a station can not be certain that no other station is transmitting. In this case, collision occurs, after a collision if it is detected during the transmission, transmission stops immediately and a brief jamming signal is transmitted to assure that all stations know that there has been a collision. After transmitting that signal, a station waits a random amount of time and then attempts to transmit. The advantage of CSMA/CD is simplicity and the disadvantage is the large number of collisions under heavy communication loads [Ref. 2]. This medium access tecnique is feasible for Turkish Battleships, because it is simple and the load at any time will not be very heavy.

2. **Control Token**

Control token is used with either bus or ring networks. The control token is passed from one station to another. One station may only transmit when it is in possesion of token. Control token is more complex and expensive technique than CSMD/CD at light loads and has better performance at heavier loads.

3. **Slotted Ring**

This method is used with a ring network. The ring is initialized to contain a fixed number of binary digits by a special node in the ring. The stream of bits continuously circulates around the ring. All slots are marked as full or empty. When a station wishes to transmit a frame, it first waits until an empty slot is detected. It then marks the slot as full and indicates the source and destination addresses. The main disadvantages are special monitor node is required and each slot can only carry a limited amount of useful information. Under normal conditions multiple slots are required.

E. NETWORK CONFIGURATION

In our Turkish Battleships model there will be seven computers linked together and a server. The IBM PC network program allows for one of four different types of configurations for each work station. There are also three levels of user configuration and one configuration for servers. The user configurations are messenger, receiver and redirector. The server configuration allows the user to employ its fixed disk and printer. The PC network will let the ship personel use some application programs such as Lotus 123, WordPerfect, Dbase III +, TCG. MF. Cakmak (presented in Chapter 4 with the user manual and evaluation). Although, in our model each PC has a fixed disk they still can use the programs in the server's fixed disk. The server allows us to employ the programs and other resources. One server may function as file server while it functions as print server. The network file server software is usually stored on a harddisk drive controlled by the file server. A file server should contain disk spaces that are accessible to other PCs on the network. A print server has a printer or another printing device that can be shared by any PC in the network.

Any file server on the file server's disk can be read and copied by any other PC in the network. The file server also maintains the control of file access. Military applications may require more restrictions on which the files are to be accessed. The file server may be a password operated one to ensure that only authorized personel can access certain files. Print server allows the user to share the printer on the network. When a printer is shared among users, they send their files to the print serving computer. Print

server saves these files in a spool file, and then transmits the file from the spool file to printer's buffer. The queuing manager program is built into the PC network software.

1. Memory and Disk Requirements

In order to run network programs, a server should have at least 320 kbytes of main memory (RAM) and at least one fixed disk. The fixed disk should be large enough to hold all the system and application software. Generally a server runs on a 80286 microprocessor based IBM PC/AT with at least 512 Kbytes of Random Access Memory. The transfer rates to the fixed disk of the IBM PC/ATs are a lot faster than that of IBM PC/XT. A server providing printer service, transfers data and files through a number of buffers. The buffers are set by default at 16K bytes of memory, but it also can be set upto 48K bytes. In our model, as stated before we will have 80386 microprocessors based computers as file and print servers. The clock can be upto 33 Mhz which is a lot faster than a regular IBM PC/AT working at 8 or 10 MHz. The Random Access Memory can be expandable to 16 Mbytes. Because of the Virtual Memory expansion, we can start with 640 Kbytes of main memory which is the memory limit for DOS 3.30. However, the future software and operating systems will require expanding the memory to 4 to 5 Mbytes.

2. Operating System

A local area network consists of hardware such as cables, adapter card, computers connected to the system, and the software. In this point, the distinction between the application software and the system software becomes an important issue. Most of the application programs will run on the local area network without giving any problem. System software and its most important joint, the operating system should be capable to meet the network's requirements. IBM introduced local area networks and the DOS 3.10 in 1985. DOS 3.10 or any greater versions of the DOS is capable to run the local area networks. The proposed operating system for Turkish Battleships is DOS 3.30 which will be upgraded to OS/2 and OS/3 as required later.

3. Users

A user uses a server's disk, directory, printer or other resources. As stated earlier each user configuration has different functions. A messenger can do everthing a server can do, except share disks, directories and printers. The messenger must have at least 256 Kbytes of memory. The receiver configuration is more limited. The receiver needs at most 192 Kbytes of main memory. It is the basic network device type. The redirector is the most limited one in capabilities. It needs 128K bytes of memory.

4. Transmission Medium

Coaxial cable (75 ohm) is used for the transmission. It allows high speed data transmission rates and the error rate is very low.

F. IMPLEMENTATION

The hardware for the attachment of any PC to the local area network typically consists of an adaptor card that is installed into the personal computer. It provides a high speed interface to the PC's internal bus. In the broadband PC networks, the main part is the frequency translator. The PCs communicate with each other by sending and receving signals. This signal traffic is managed by the frequency translator. From the translator, the cable is connected to the two way splitter. One branch from two way splitter leads to eight-way splitter. The PCs are connected to the eight way splitter. If there is no more than eigth PCs in the network, just one eigth splitter will be enough. In this case, we will not use the other branch of the two-way splitter. Our model will not have more than eight PC, so one eight-way splitter will be good for our model. If we had more than eight PCs, then we would also need multiple clusters.

In the configuration, there are two basic types of devices. These are servers and non-servers. We will configure all non-servers as messengers. This will provide the better features. In this type of configuration, the PCs can use server's file space, directory or printer; they can receive and transmit messages.

G. INTEL 80386 MICROPROCESSOR ARCHITECTURE

Intel 80386 microprocessor is designed to handle mainframe-size applications with typical data structures, large integers and large number of programs. It can address 4 gigabytes of physical memory and supports virtual memory. The concept of virtual memory was first pioneered by mainframe architectures [Ref. 3]. Virtual Memory allows the use of programs and data structures larger than the actual physical memory. Its 64 terabyte virtual memory capacity lets the processor handle even the largest programs and data structures. The processor's full 32 bit architecture provides a high degree of parallelism and a fast local bus. The eight general purpose registers can be used as accumulators, for logical operators, or as addressing and data registers. Each register can be used for 32 or 16 bit values, four of these registers can also be split into pairs of 8-bit registers. There are six 16-bit segment registers to implement the memory segmentation. If the 80387 Math-coprocessor is used, eight 80 bit floating point registers are available. Operating system makes use of a number of registers and these are invisible to the application programs.

The 80386 provides object code compatibility with programs created for Intel's iAPX processor family 8088, 8086, 80186, 80188 and 80286. The 80386's virtual 8086 facilities let MS-DOS based 8086 programs run concurrently with programs written for 32-bit operating systems, with each program operating as a separate task in a protected multitasking form.

A complete set of instructions, implements the processing of data types. The instruction set is a superset of 8086 and 80286 instructions. Instructions average 3.1 bytes in length and require 4.4 clock cycles for execution when measured in a typical application set. One of the most important features is that 80386 is partitioned into a six-stage pipelined architecture. Each of these units can work independently and in parallel with the others. This also enables the chip to overlap execution of multiple instructions. The paging and segmentation units are the part of the pipeline as well. Together, they build up the processor's memory management unit (MMU). The 80386 local bus achieves an overall throughput of 32 megabytes per second at 16 Mhz [Ref. 4].

1. Cache Strategy

The average sizes of the caches are about 32 Kbytes for off-chip caches. The hit rates are in 90 to 99 percent range. The cache size looks very small for providing significant benefits, but this size is optimized for 80386 off-chip caching. On-chip caches have typically hit rates of only 20 to 30 percent. Dynamic bus sizing allows transfers of 16 or 32 bit data. The processor can change the dynamic bus size during the execution and the result is completely transparent to the application programmer.

2. Memory Management

Virtual memory defines a logical address space that can exceed the physical address space. Logical addresses pass through translation hardware to yield the physical address. The large logical address space is subdivided into units of allocation, segments or pages. Segments and pages can be swapped into and out of main memory from secondary storage devices. This swapping into and out of the main memory process is also transparent to the application programmer. The 80386 makes use of both paging and segmentation schemes taking advantages of both strategies. In segmentation, logical memory is subdivided into segments of variable length. Segments in 80386 can be any size up to 4 gigabytes. In the segmentation process, address translation is managed by descriptor table and page-table entries. This architectural design is very common in all Virtual Memory implementations. Almost all virtual memory managements are strictly based on the dynamic address translation from logical address to physical address space. Paging is a little bit different than segmentation in 80386. Pages are small blocks of fixed

size. The 80386 page lenghts are 4K bytes. Paging simplifies the swapping algorithm by providing uniform units of allocation. Paging is used to manage the physical memory.

The 80386 has combined the use of segmentation and paging. This feature makes the address translation very fast.

3. Multitasking

The 80386 was designed for high throughput multitasking. Each task sees only it's own execution context and is unaware of other tasks running concurrently.

4. Software Compatibility

The 80386 supplies binary code compatibility with all previous 8086 family software applications. We can port operating systems and programming tools directly to the 80386 without recompilation or assembly. For 8086 programs, the processor provides virtual 8086 mode, which establishes a task-selectable protected 8086 environment within the multitasking framework. In this environment, tasks can take advantage of the 80386's protection and paging mechanisms, while running the non-privileged 8086 instructions directly at roughly 16 times the speed of an 8086.

5. Performance and Conclusion

The on-chip Memory Management Unit, a fast local bus, pipelining and the processor's compact code contribute to a fast throughput. Recent benchmark tests set the 80386's performance at 3.64 MIPS, averaged over 14 applications and 31 million instructions. The chip's interrupt response has been clocked at 3.7 microseconds from event to first handler instruction. The 80386's performance can also be increased with an 80387 coprocessor.

The tests have proved that the performance of the 80386 is compatible with many mainframe type computer. Multitasking, paging and segmentation, software and hardware compatibility, speed and pipelined architectures are very much satisfying. These extended features and affordable price difference between Intel 80286 and 80386 has encouraged us to employ 80386 microprocessors in the proposed Turkish Battleships model.

H. LOCATION IN THE SHIP

We will have seven PCs and a server on the ship. The PCs will be located in the following locations as shown in Figure 1.

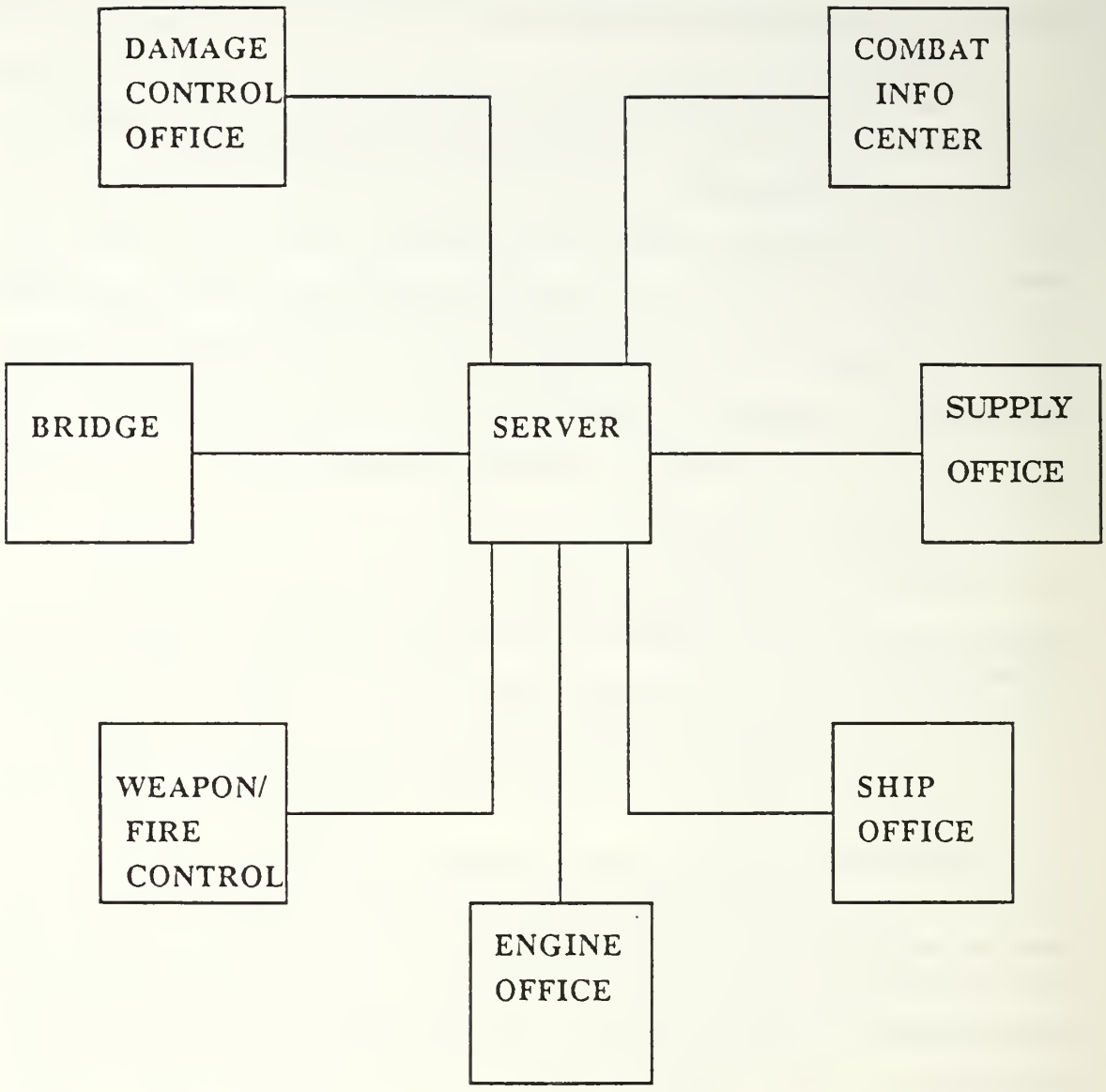


Figure 1. PC Locations in the Battleship.

1. Damage Control Office,
2. Engine Office,
3. Weapon / Fire Control Room,
4. Supply Office,

5. Ship Office,
6. Bridge,
7. Combat Information Center.

The server can be located to the Ship Office. We will also have additional monitors on the bridge and Commanding Officer Office. This will let the Commanding Officer or the Navigation Duty Officer see the radar screen and Combat Information Center solutions at the same time. If any ship department needs a PC, our server can still support the system. The ship's office should also have a modem in its PC to communicate with headquarters. We will support the system with two printers. One of them will be laser printer, and the other one is a dot matrix printer. They both will be installed in the ship's office. The dot matrix printers are quite fast and inexpensive. They can be used for regular type applications in the ship. They can be used for regular type applications in the ship.

III. THE SOFTWARE IMPLEMENTATION

This Chapter states the lack of military software problem and then recommends a solution. It also evaluates and statisticaly analyses the software package created for Turkish Battleships.

A. DEFINITION OF THE PROBLEM

Software has a large and increasing effect. Its cost has been increasing as well. This increasing effect and cost directly reflect the military applications. Because of the security aspect and the special goals of the military applications, it is even harder to implement than the civilian software. In some cases, military applications may include or use the civilian software. Thus, in terms of layers, military software can be defined as the superset of the civilian software. Generating military oriented software may require very hard steps, reaching the so-called "gold standards" of data and programs may not even be possible. The fact is that there is a great need of software in military applications at present.

B. RECOMMENDED SOLUTION

The software package TCG. MF. Cakmak is designed to meet the most important software requirements of Turkish Battleships. It mainly concentrates on three major ship applications. These are:

1. Combat Information Center,
2. Personnel Evaluations,
3. Damage Control.

The fact is that the package MF. Cakmak is an initial step to create a navy oriented software for Turkish Battleships and it will not meet all the software requirements of those ships. In the near future, we can meet these software requirements by implementing them.

C. GENERAL CHARACTERISTICS OF THE PACKAGE

1. Language and Software Compatibility

In the implementation of the package, Turbo Pascal 4.0 was employed. It allowed us to use effective graphic routines and as well as screen text mode and disk operating system library routines. The compiler also provided all well known abstract data

structures. We could obtain *.tpu (Turbo Pascal Unit) files by writing our own library routines. The main program could combine all *.tpu files and create an executable file with .exe extention. The Turbo Pascal IV.0 has all the features we would expect from a fourth generation high level language.

In spring 1988, Borland International built up Turbo Pascal 5.0. Turbo Pascal 4.0 and Turbo Pascal 5.0 are compatible, that is the next software requirements that can be implemented in Turbo Pascal 5.0. In terms of the compatibity, the language used to implement the future requirements will not make any difference as long as the compiler generates an executable code. Thus, using Ada, C, Modula-2, Turbo Pascal and even Basic does not make difference.

2. Hardware Compatibility

The TCG. MF. Cakmak will run on any IBM or compatible computer system. The package was originally designed on an IBM compatible computer working with Intel iAPx 80286 microprocessor at 8 MHz. In order to provide the best interface with the user, I have forced the program to run slower with I/O interrupts and operating system delays. The program will work approximately two times slower on a 8086 or 8088 based computer. In Turkish Battleships, the program will be working on 80386 microprocessors, three times faster than it does on a regular IBM PC/AT. The I/O interrupts and the operating system delays will still assure the proper user interface. For example, the radar sweeps will be faster on 80386 based computer.

The excess usage of the graphics routines, brings up the graphics compatability problem. The original package was designed on monochrome monitor with Hercules graphics adapter. It has been very common to use monochrome monitors in military applications. The main reason behind this, is that of the health conditions of the battle ship computer operators. A department chief or the Commanding Officer may ask a computer operator to stand by or process data for a very long time without taking break under emergency conditions. The monochrome monitors also provides a better resolution which is one of the most important factor in graphically oriented software packages. With its 720 x 348 resolution, we could obtain pictures on 250560 pixel graphic array. On the other hand, we wanted see our pictures on a EGA oriented screen with better colors. Generating proper formulas and aspect ratio relations allowed us to run our programs on EGA monitors with 640 x 350 resolution and 224000 pixel graphic array. The monochrome version of TCG. MF. Cakmak is called version 1.0 and the EGA version is called version 1.1. This thesis presents the version 1.0 at the end of Chapter 4. The size of the some files are a little bit larger in version 1.1 because of managing color

graphics. As we stated before, the version 1.0 will be employed in Turkish Battleships. The version 1.1 can be used in Naval Academy and in other educational centers for training purposes. Two different versions will be treated as install options.

3. Math-coprocessor

The package does not require math-coprocessor 8087, 80187, 80287 and 80387 on the respective computers. Having a math-coprocessor will speed up the execution though.

4. Implementation

The TCG MF. Cakmak is a software package, implemented to be used in Turkish Battleships stand alone and local area network computer systems. Although, it does not meet all the software requirements of the Turkish Battleships, it does solve the problem that is called lack of military application software. It mainly concentrates on the major navigational, scientific-calculational and administrative applications. These are CIC (Combat Information Center), Personel Evaluations and D C (Damage Control).

The main function of the CIC department in a battleship, is to get information and determine about the targets' movements. A target does not have to be an enemy surface ship, aircraft or submarine. Even a friendly navigating fishing boat is a target for CIC. Thus, CIC is responsible for getting and processing all the data about targets. This responsibility also assures the safety of the navigational traffic at sea. The other important function of the CIC is to suggest the best speed and course combination to the navigation duty officer or commanding officer. This function becomes very important expecially in national and international exercices when one pivot ship is directing all others. Without using computers, CIC problems may sometimes be very difficult and time consuming. The worst is that the problems may not be solved correctly under war or exercise conditions, because of the overtired CIC personnel. Experience has shown that almost all navigation duty officers have been witness to a wrongly solved CIC problem. Wrong CIC solutions, even in peace conditions, may cause the loss of life and other values. The CIC part of the TCG. MF. Cakmak does correctly solve both type of CIC solutions a lot faster than the traditional CIC methods do. It is also a real time application. The computer aided CIC implementation can track 30 targets at a time. The CIC in the ship never processes data concerning more than five targets at a time.

The second part of the package is personnel evaluations. The reason we implemented it is that the personnel evaluations in the navy has not been very effective and we wanted to improve the present methods. The evaluations are done by department

heads once a year. The Personnel Evaluations implementation allows a department head to evaluate the people working under his command more than once a year and easily. This also prevents the evaluation of people under physicological pressure at that specific time. This part of the code has also been a real time application.

The Damage Control is a very important function in a battleship both in peace time and war conditions. The damage control branch, is a subdivision of the engineering department in battleships. Damage control personnel under the command of the damage control officer is responsible for the ship stability, fire fighting, nuclear, biological and chemistry warship readiness, providing of watertightness of the ship, and every kind of engineering application. The branch is also responsible for the maintanence of the fuel-oil, diesel-oil, feed and potable water tanks. As a former Damage Control Officer, I implemented seven different functions in the damage control part. It will allow damage control officers and assistants to work more efficiently.

D. STATISTICAL ANALYSIS

As software systems gets larger, reliability requirement gets more stringent. A statistical method can be used to explain this fact. A conventionally designed software system fails when one of its module fails [Ref. 5]. Let us define:

N: The number of modules in a software system.

P: The probability that a typical module is free from faults.

P^N : The probability that the entire system is free from faults.

1 - P : The probabily that a typical module has a fault.

Q = 1 - P.

$P^N = (1 - Q)^N$ = 1 - N.Q + higher order terms using a binomial expansion:

P^N = 1 - N.Q.

1 - P^N = N.Q the probability the entire system has a fault.

In Our TCG. MF. Cakmak implementation, there are 20 modules. Even if each module were 99 percent error free, the reliability of the total system would be 81.2 percent which may not be allowed in a military application. The same approach can be used to analyse each module's reliability, because each module consists of a number of procedures and functions. The largest module, in TCG. MF. Cakmak implementation is Combat Information Center (CIC). The CIC has a total of 43 procedures and functions. If each procedure and function were 99 percent error free, than the reliabiliy of the CIC would be 64.9 percent. We could never run CIC program in our ships with this reliability

percentage, otherwise we would have to give up thousands of dollars a battle ship and hundreds of people.

The TCG. MF. Cakmak has worked properly in testings. The problems may be caused by user faults (i.e. entering wrong data) and these faults can be best prevented by special short term courses.

The CIC program solves both type of problems 100 percent correctly as long as the data entry is correct. The TCG. MF. Cakmak has 20 modules, additionally it uses four library routines. These are Graph.doc, Crt.doc, Dos.doc and Hercules.bgi (or Egavga.bgi on EGA oriented monitors). The total source code is 131 K bytes and the generated executable code is 105 Kbytes. The program consists of 4310 lines of code.

IV. USER MANUAL

This chapter provides the user manual for the effective use of the software package. At the end of the chapter we presented the source code.

A. GETTING STARTED WITH TCG. MF. CAKMAK

The sign on is password operated. In order to be able to log on, the users must have an identification number and a password. After running the executable code the program draws a ship picture and writes the name of the ship giving the message "Press Enter to Continue". If no one is using the program, this picture will stay on the screen.

If the user hits the carriage return at this initial screen mode, the program next asks the user to enter the identification number followed by the password. If identification number is already not an acceptable one, then the password entry will never be asked. In this case, the program will display the message "User Authorization Failure ". One failure in password or identification number entries, causes the program to ask these two entries two times. This prevents any coincident password break and logging on to the system.

After logging on, the program will ask the user what he/she wants to do. As stated before, the program displays three options at this stage. These are:

1. Combat Information Center.
2. Personnel Evaluations.
3. Damage Control Center.

TCG. MF. Cakmak menu options are shown in Figure 2. The user can enter his/her option by pressing the related number. An unrelated entry at this moment will cause the program to give the following message:

" Invalid entry, the feature not implemented yet "

" Do you want to quit TCG. M.F. Cakmak ?"

An entry of "y" or "Y" will let the user to get out the program environment. If the user enters option "1", the program will call the CIC program and run it.

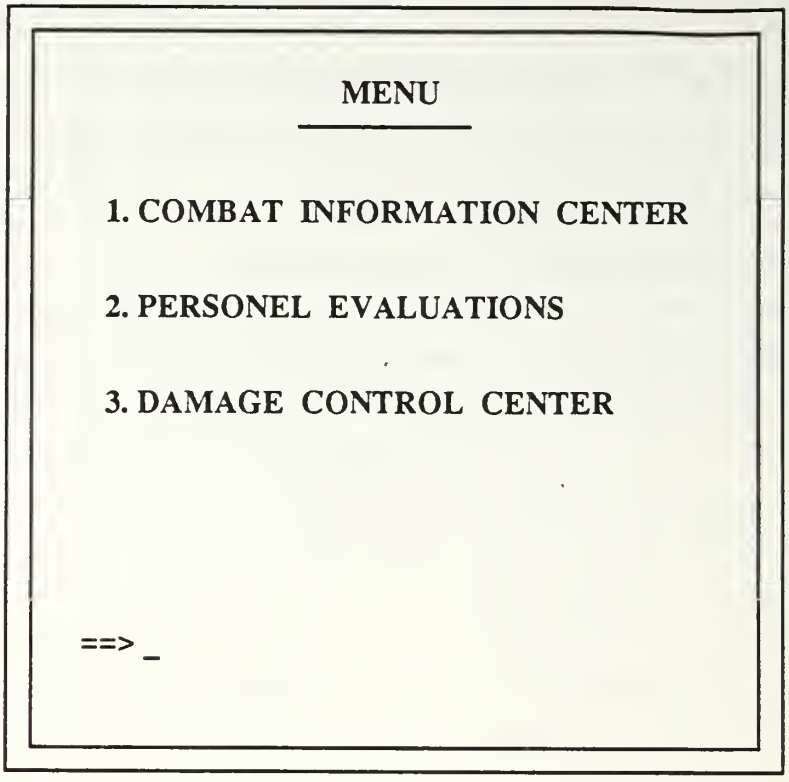


Figure 2. Menu Options in TCG. MF. Cakmak.

B. COMBAT INFORMATION CENTER

The CIC program first draws the CIC data entry table on the screen, in text mode. The screen and the table is shown in Figure 3. The program gets the time and date information from the operating system and displays it in the CIC entry table. If the user starts solving any kind of CIC problem, he she should first hit the carriage return, this will take the take cursor from the time cell down to OwnShip PivotShip cell. The users acceptable entries can be:

1. o, O or carriage return for ownship type calculations,
2. p or P for pivotship type calculations,
3. q or Q for quitting the program.

COMBAT INFO CENTER			
OWN SHIP DATA		TARGET SHIP DATA	
TCG.M.CAKMAK	STATUS .A.	RELATIVE BEARING	
TODAY	MON. 7/25/1988	RANGE	
CURRENT TIME	11:55:18:78	AT TIME	
SPEED	12 Knots	RELATIVE BEARING	
COURSE	120 °	RANGE	
OWNSHIP / PIVOT	o	AT TIME	
CHANGE FUNCTION		TARGET SPEED	
MORE SHIPS		TARGET COURSE	
TARGET TRACK.		CPA	

Figure 3. CIC Data Entry Form.

If the user wants to do first type of (Ownship type calculations) then he/she enters either o, or O, or the user can just hits the carriage return. This will initialize the program. In this case the computer first enters the ownship data in ownship data cells which are ownship speed and ownship course. When the radar operator is ready to present the target ship data at time t1, the computer operator first hits the carriage return to move the cursor from time counting cell to the target ship data cells.

The data coming from the radar through radar operator is easily entered to respective cells in the table. After finishing data entry for any specific target, the computer will remind the computer operator that it was the first target tracked.

In this stage, the cursor immediately moves to the MORE SHIPS cell. An entry of y or the Y will let the user enter all targets data coming out from the radar and other sensors. When the second positions of the targets are reported by the radar operator, the computer operator will simultaneously enter these data as well. After entering the secondary positions of the targets, pressing the carriage return will solve the problem for the first target. If there is more than one target, then the computer operator can enter y or Y when the program asks whether the computer operator wants to see the respective target's true speed and course values.

During the solution and the exhibition of the CIC problem, if the closest point of approach is less than 500 yards, then the computer will display sound and vision effects. These are a flashing "Collision" message and a warning sound and signal. If the target is moving faster than 100.000 yards hour (50 mph) then the computer will determine that the target is probably an air target. Air target alarm will be displayed on the screen as a flashing "AIR".

After solving the problem, the computer operator can solve another problem by typing y in more ships cell. If the user wants to see the radar screen, he can type r when the cursor flashing at change function cell. The radar screen is quite useful for the radar operator, because it can display all the target information in true speed and true course vectorial form. This implementation is more informative than an ordinary radar screen, because every spot seen on the radar screen is a function of ownship speed and course. Thus, nothing on the radar screen is real, but all the information is relative. The computer screen radar information with the real values of the targets' data can also be transmitted to the monitor located in the Commanding Officer's room. This lets the CO be able to see the tactical scene, if he is not even at the bridge. A sample radar screen and solution is illustrated in Figure 4. Either CO or the computer operator, can get out of the radar shell just by hitting the carriage return.

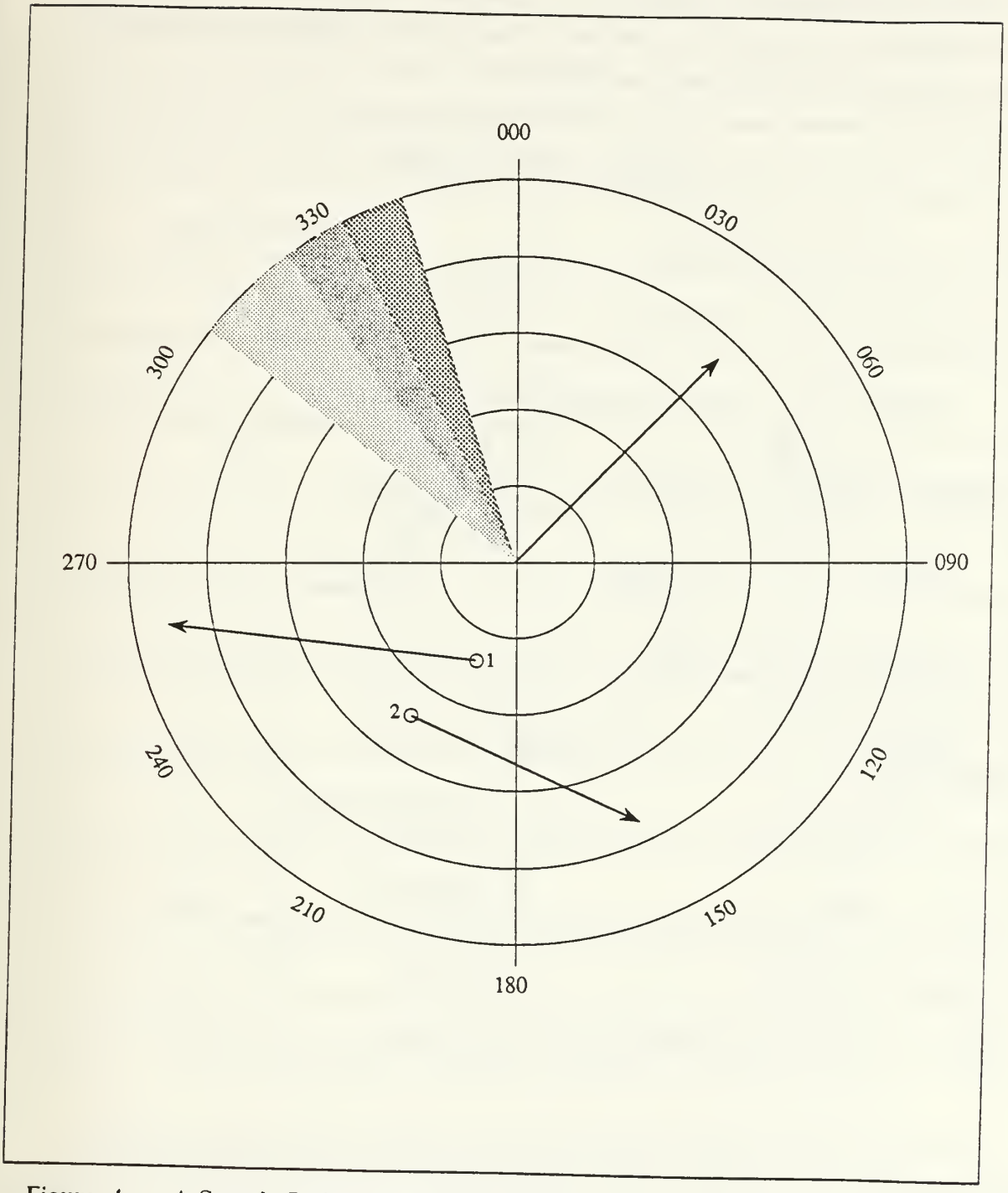


Figure 4. A Sample Radar Screen and Solution.

The pivot ship solution is quite similar to the one we explained in ownship target tracking solutions. The difference is that the CIC entry table is changed. The CIC entry table for pivot ship solution is shown in Figure 5.

COMBAT INFO CENTER			
PIVOT SHIP DATA		OWN SHIP DATA	
TCG.M.CAKMAK	STATUS .A.	RELATIVE BEARING	
TODAY	MON. 7/25/1988	RANGE	
CURRENT TIME	11:55:18:78	AT TIME	
SPEED	12 Knots	RELATIVE BEARING	
COURSE	120 °	RANGE	
OWNSHIP / PIVOT	p	AT TIME	
CHANGE FUNCTION		OWNSHIP SPEED	
MORE SHIPS		COURSE	
TARGET TRACK.		CPA	

Figure 5. CIC Data Entry Table for Pivotship Solution.

The computer operator gets the pivot ship data from the radar operator or the pivot ship itself reports its speed and course. Then the computer operator first enters the current positions (range and relative bearing which are both relative to the pivot ship) and then the position where the pivot or the commanding ship wanted us to get to. The operator can also enter the time difference allowed to get to the specific position. Hitting a carriage return will allow the user to see the solution of the problem.

This solution is immediately presented to the bridge and navigation duty officer. The operator can go back to the radar screen or can quit the program. The operator can even do an ownship target tracking solution if he wants to. The Fig. 6 shows the finite state automata model of the possible states that the CIC program can have at any given time. It also shows transitions between allowed states.

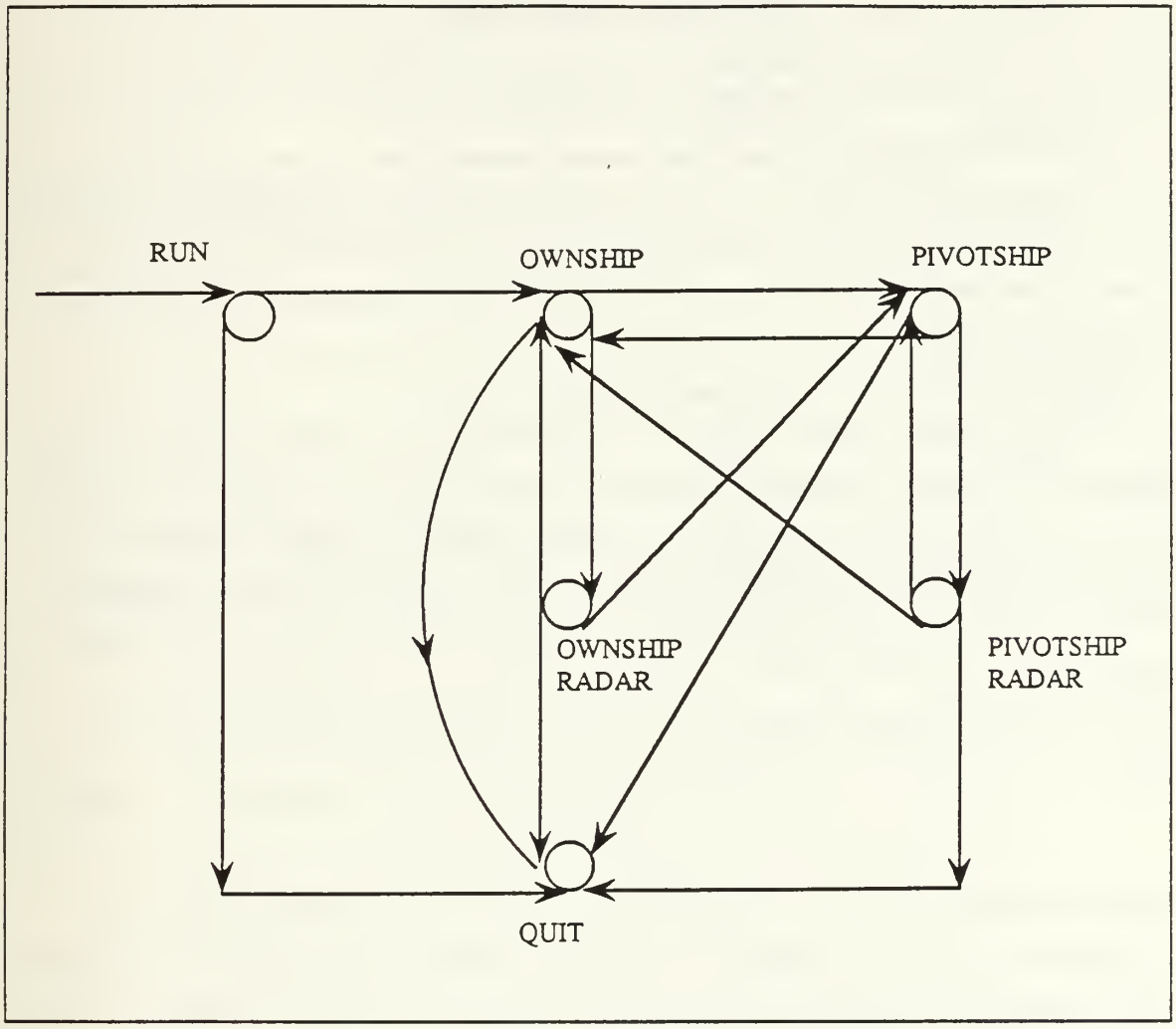


Figure 6. Finite State Automata Model for CIC States.

The program takes the user out of the executing CIC shell, if the user enters the q at the Change Function cell. When q entered, the program will display a "Thank You For Using CIC" message.

C. PERSONNEL EVALUATIONS

If the user wants to evaluate any person, he she can enter 2 in the main menu. The TCG. MF. Cakmak program will ask the following questions to the user

Q1. Enter the last name of the person to be evaluated.

The user enters the last name of the person to be evaluated.

Q2. Enter the firstname of the person to be evaluated.

The user enters the firstname of the person to be evaluated.

Q3. Enter the rank of the person to be evaluated.

The user enters the rank of the person to be evaluated.

The program then looks for the previous evaluation file and data for this specific person to be evaluated. If the program finds the data in the secondary memory device, it loads this data into the main memory; if it can not find the data and the file, then asks the following question.

"The file does not exist, do you want to create a file ?"

In order to evaluate anyone, the user has to create a file. An entry of 'y' or 'Y' will let the program create a file for the person to be evaluated. At this stage of the execution, the program draws the personnel evaluation table. The Figure 7 shows the general structure of the personnel evaluation table. The user starts entering the grades in respective cells. Each cell has to be filled out with some integer (or real) number representing the grade for this cell. After entering the last grade, the program calculates the total and the average values, and writes in the date and the time when the grades were assigned.

PERSONNEL EVALUATIONS				
REALIBILITY	10	10	JOHNSON	SAVE "S"
RESPONSIBILITY	10	10	GEORGE	MODIFY 2 X CR.
LEADERSHIP	9	9	LT J.G.	QUIT ^Z
COMPREHENSIBILITY	10	9		
STAMINA	9	10		
PERSONALITY	10	10		
GOOD MORAL	10	10	DATE	15_11_1988
PRODUCTIVITY	10	9	TIME	13:17:58
GENERAL BEHAVIOUR	10	10	TOTAL	97
PROFESSIONALITY	9	9	AVERAGE	9.70

Figure 7. Personnel Evaluations Data Entry Form.

The user then enters the lastname and a dot followed by the first three characters of the person who has been evaluated. The data and file can be saved by hitting the s or S. The entries can also be modified by bringing the cursor to the respective cell. The user can easily get out of the personnel evaluations shell by simply typing the Cntl-Z. The program asks :

"Do you want to evaluate another person ?". The user can enter y, Y, n or N. If he/she does not want to evaluate another person then program asks

"Do you want to quit TCG. MF. Cakmak ?". An entry of y or Y will take the user to the disk operating system environment.

D. DAMAGE CONTROL

The damage control program menus are wrap-around, pull-down menu type. Because of having more options, we thought this kind of menu would be easier to use for the users of this program. In this part of the package, there are seven executable functions and the last function (function eight) is the exit function. When selected, the damage control program will present the following options to the user:

1. General Ship Information,
2. Draft Diagram and Functions,
3. Show Ballast Tanks,
4. Show Fuel Tanks,
5. Show Water Tanks,
6. Fire Fighting Exercise,
7. Liquid Loading Diagram,
8. Exit.

Any option can be selected by using Up and Down Arrows or entering the respective number. For example an entry of '3' will cause the program to show the ballast tanks.

1. General Ship Information

This program draws the 17 watertight sections of the ship which are located under the main deck. It also displays all the tecniqual information about the ship as well as the selected frame numbers of some watertight compartments.

2. Draft Diagram and Functions.

The user enters the forward and the after draft marks of the ship into the respective entry cells. The program calculates the displacement (in tons) and the immersion (in tons/inch) graphically. The program lets the damage control officer or assistants figure out what the displacement and the immersion are at any time. The displacement and the immersion are the major factors effecting the ship stability. The carriage return will take the user back to the damage control menu.

3. Tank Diagrams

The TCG. MF. Cakmak program allows the damage control personnel to see the ballast, fuel oil, diesel oil, feed water and potable water tanks of the ship. In order to get back to the damage control menu, the user simply hits the carriage return.

4. Fire Fighting Exercise

This portion of the damage control implementation was designed to use both in real fire and in fire exercices. The user enters the type of the fire. If the fire is 'A' class

fire (regular fire) then the user next enters the location of the fire. A location in the ship can be defined in terms of deck number and the frame number. After entering the deck and the frame numbers, the program displays the available closest fire stations for use. It also flashes the message 'turn off the ventilation' and the responsible repair team for extinguishing the fire. The program at the same time records the time spent to take the fire under control while reminding the communication line 2JZ has to be activated. The program, as stated before can be used in fire exercises as well to assign grades to each of the repair teams. The time spent to control the fire is the score that the specific team makes. The program is also designed for use in case of 'B' class fire (electrical fire). In case of B class fire, it displays following flashing messages :

1. Do Not Use Water,
2. Use CO2 or Dry Chemical Agents,
3. Communication Line is 2JZ,
4. Score.

The time spent to control the fire is again the score of that specific repair team. After first use of this program, fire fighting implementation will require two more confirmation, if any other damage control program has not been used after the fire fighting program.

5. Liquid Loading Diagram

The liquid loading diagram gives information about any fuel oil, diesel oil, ballast, feed or water tank. The information about any tank consists of following techniqual data :

1. Capacity of the tank,
2. The effect to the forward draft,
3. The effect to the after draft,
4. The effect to the trim,
5. The location of the tank in terms of the frame numbers.

The user of the program can easily get all the information about any tank of the ship. This allows the damage control duty assistants to arrange the 0 degree trim and a good stability conditions. The entry to the program is just the name of the tank, but in upper case letters. Because all tank names are described with upper case letters in the navy. Otherwise, the program will give the message "No such tank, do you want to

continue ?". After getting all the information about the tank, the user can go back to the damage control menu simply by hitting the carriage return.

6. Exit

This last function is used to leave the damage control program. When used, The MF. Cakmak implementation will ask the following question:

"Do you want to quit the MF. Cakmak ?".

An entry of y or Y will take the user back to the disk operating system environment.

E. SOURCE CODE

```
    program M.F._CAKMAK;
[*************************************************************************
 *************************************************************************

                        TCG. MF.CAKMAK 1.0
                        TURBO PASCAL 4.0.
                          3.26.1988
 *************************************************************************
 *************************************************************************]
    uses Graph,Crt,Dos,Testgr,Try2,UserAuth,DrawShip,outmenu,perfile,cic2,dm8;
    label 50;
    var
     TestGrOk,Authorized:Boolean;
    begin
     50: InitGraph(GraphDriver,GraphMode,' A:drivers'); [** Initialize Graphics **]
    [*** Test whether proper graphics devices are already installed *** ]
      TestGrOk:= Test_Graph_Device(GraphDriver,ErrorCode,GraphMode);
      if TestGrOk then
      begin
      [ If installed then go ahead ]
 Draw_Ship;    [ ** Draws the Start_up Screen ** ]
         Authorized:= User_Authorization(Users);
end;
   while not Authorized do
 begin
  TestGrOk:= Test_Graph_Device(GraphDriver,ErrorCode,GraphMode);
  if TestGrOk then begin
  Draw_Ship;
  Screen_handler;
  Authorized:= UserAuth.User_Authorization(Users);
  end;
  end;
  CloseGraph;
```

```
    Clrscr;
outmenu.out;
       writeln('DO YOU WANT TO QUIT M.F.CAKMAK . . ?');
Ch := ReadKey;
  if ((Ch = #78) or (Ch = #110)) then
  goto 50 ;
  end.
```

```
unit DrawShip;
[*************************************************
   This Unit draws the ship picture on the screen.
   The unit is called by the main program.
 *************************************************]
interface
uses Graph,Dos,Crt,Testgr;
Procedure Draw_Ship;
implementation
 Procedure Draw_Ship;
  const
     ship: array (.1..5.) of PointType = ((x:150;y:250),
                                          (x:175;y:275),
                                          (x:545;y:275),
                                          (x:560;y:250),
                                          (x:150;y:250));
            [*** SUPERSTRUCTURE COORDINATES ***]

      deck: array (.1..4.) of PointType = ((x:225;y:250),
                                           (x:250;y:225),
                                           (x:450;y:225),
                                           (x:450;y:250));
             [*** MISSILE LAUNCHERS COORDINATES ***]

launcher: array (.1..5.) of PointType = ((x:470;y:210),
                                         (x:490;y:210),
                                         (x:510;y:250),
                                         (x:490; y:250),
                                         (x:470; y:210));

            [*** FORWARD GUN COORDINATES ***]

 gun : array (.1..5.) of PointType = (( x:205;y:235),
                                     ( x:190;y:235),
                                     ( x:175;y:250),
```

```
                                    ( x:205;y:235));
                [*** TARET COORDINATES ***]
  taret : array (.1..5.) of PointType = (( x:182;y:243),
                                    ( x:184;y:241),
                                    ( x:165;y:235),
                                    ( x:163;y:237),
                                    ( x:182;y:243)),
     radius = 10;
    var
      Y1:Integer;
         begin
          Rectangle(100,25,(GetMaxX-100),(GetMaxY-175));
          SetTextJustify(CenterText,CenterText);
          SetTextStyle(1,Horizdir,4);
          Y1:= GetMaxY-175;
OutTextXY((GetMaxX-100)div2+50,Y1div2,'MARESAL FEVZI CAKMAK');
          SetTextStyle(1,Horizdir,2);
          OutTextXY(150,325,'Press Enter to continue...');
  [ *** DRAW THE SHIP *** ]
  DrawPoly(SizeOf(Ship) div SizeOf(PointType),Ship);
  FillPoly(SizeOf(Ship) div SizeOf(PointType),Ship);
  DrawPoly(SizeOf(Deck) div SizeOf(PointType),Deck);
  FillPoly(SizeOf(Deck) div SizeOf(PointType),Deck);
  Drawpoly(SizeOf(Launcher) div SizeOf(PointType),Launcher);
  FillPoly(10,Launcher);
  DrawPoly(SizeOf(Gun) div SizeOf(pointType),Gun);
  FillPoly(3,Gun);
  DrawPoly(SizeOf(Taret) div SizeOf(PointType),Taret);
  Arc(300,210,240,60,radius); [ radar ]
  Line(307,216,307,250); [ top radar waveguide ]
  Line(370,225,370,180); [forward radio antenna ]
  Line(465,250,465,190); [backward radio antenna ]
  Line(560,250,560,245); [ aft prevent ]
  Line(520,250,520,245); [ forward prevent ]
  Line(520,245,560,245);
```

```
Line(530,250,530,245);
Line(540,250,540,245);
Line(550,250,550,245);
Line(560,245,520,245);
Line(50,270,650,270);
Line(560,242,520,242);
Line(132,295,148,295);
Line(458,319,477,319);
[  SetTextStyle(2,Horizdir,1);
LowVideo;
OutTextXY(200,60,'351'); ]
readln;
end;
end.
```

unit outmenu;

[**********]

This is the main unit in TCG. MF. Cakmak. The module handles all the user options. The module calls Cic2, dm8, and perfile modules respectively. The unit is called by the TCG. MF. Cakmak.

[**********]

```
    interface
    uses menu,Crt,perfile,cic2,dm8;
    var  User_Choice : char;
    procedure out;
    implementation
    procedure out;
    begin
    ClrScr;
    menu.menu_window(User_Choice);Clrscr;
     window(1,1,80,25);
     if User_Choice = #50 then  beginClrscr;
 perfile.main;
 clrscr;
 end
 else if User_Choice = #49 then begin
    clrscr;
    cic2.main;Clrscr;
    end
    else if User_Choice = #51 then begin
    clrscr;
    dm8.main;
    end
   else begin
    clrscr;
    writeln('INVALID CHOICE , NOT IMPLEMENTED YET ');
    end;
end;
```

```
unit menu;
{*********************************************************************}
 This module is called by the Turbo Pascal Unit outmenu. It is used
 to draw a menu window and to display the menu options.
{*********************************************************************}
    interface
    uses Crt;
    const
    LeftCorner = #201;
    HorzBar = #205;
    RightCorner = #187;
     VertBar = #186;
     LowLeftCorner = #200;
     LowRightCorner = #188;
     X1 = 20; Y1 = 5;
     X2 = 60; Y2 = 21;
    var i : integer;
    procedure menu_window(var User_Choise : Char);
     implementation
     procedure menu_window(var User_Choise : char );
     begin
      Window(X1-1,Y1-1,X2 + 1,Y2 + 1);
      ClrScr;
      Window(1,1,80,25);
      gotoxy(X1-1,Y1-1);
      write(UpLeftCorner);
    for i:= X1 to X2 do
    Write(HorzBar);
      write(UpRightCorner);
       for i:= Y1 to Y2 do begin
        gotoxy(X1-1,i);write(VertBar);
        gotoxy(X2 + 1,i);write(Vertbar);
        end;
        gotoxy(X1-1,Y2 + 1);
   write(LowLeftCorner);
```

```
   for i:= X1 to X2 do
      write(HorzBar);
      write(LowRightCorner);
      window(X1,Y1,X2,Y2);
   gotoxy(16,1); writeln(' M E N U ');
   gotoxy(15,2);
   for i := 15 to 25 do
       write(#196);
gotoxy(2,4); writeln('          1. COMBAT INFO CENTER ');
gotoxy(2,9);  writeln('          2. PERSONEL EVALUATION ');
gotoxy(2,14); writeln('          3. DAMAGE CONTROL CENTER ');
gotoxy(2,20); write(' = = > ');
gotoxy(7,20); Readln(User_Choise);
delay(1500);
ClrScr;
 end;
 end.
```

```
unit try2;
{*************************************************************************
```

This module is called by main program and it is used to set the graphics mode of the computer if the computer has proper hardware The program also calls UserAuth module to check the authorization.

```
*************************************************************************}
  interface
  uses Graph,Testgr,UserAuth;
  procedure screen_handler;
  implementation
    procedure screen_handler;
     var Gr,Found:Boolean;
      begin
        InitGraph(GraphDriver,GraphMode,'');
        Gr:= Test_Graph_Device(GraphDriver,ErrorCode,GraphMode);
          if Gr then
             begin
               Found:= UserAuth.User_Authorization(Users);
               if Found Then
                        begin
               SetGraphMode(GraphMode);
               end;
       end
          else
             CloseGraph;
       end;
     end.
```

```
unit userauth;
[*************************************************************************]
   [* This  unit is used to check  whether the user  has authorization or not. The Unit
is called by  the main program. This module has function called as User_Authorization
which turns out to be true or false boolean values. *]
 *************************************************************************]
    interface
   uses Graph,Crt,Dos,Testgr;
    type
    User = record
     ID: Integer;
     Password: String;
     end;
     UserList =  array (.1..10.) of User;
     var
     Users:UserList;
     i:integer;
     function User_Authorization(Users:UserList):Boolean;
     implementation
     function User_Authorization(Users:UserList):Boolean;
     Var Temp_ID:Integer;
          i:integer;
     Temp_Password:String;
     begin
     if Test_Graph_Device(GraphDriver,ErrorCode,GraphMOde) then begin
       Users[ 1 ].Password:= 'bicir';
       Users[ 2 ].Password:= 'aysegul';
       OutTextXY(150,50,'ID NUMBER: ');
       readln(Temp_ID);
       if ((Temp_ID > = 1001 ) and (Temp_ID < = 1010 )) then
          begin
           OutTextXY(150,100,'PASSWORD:');
             readln(Temp_Password);
             if Temp_Password= Users[ Temp_ID-1000 ].Password then
               begin
```

```
          User_Authorization:= True;
              for i:=0 to 14 do
            OutTextXY(270,150+(i*10),'ACCESS SUCCESFUL');
            end
         else
             begin
             OutTextXY(150,50,'          ');
           OutTextXY(150,100,'PASSWORD:');
          readln(Temp_Password);
           If Users[ Temp_ID-1000 ].Password= Temp_Password then
              begin
               for i:=1 to 14 do
             OutTextXY(150+(i*10),150+(i*10),'ACCESS SUCCESSFUL');
                User_AutHorization:= True;
              end
              else
             begin
              OutTextXY(200,200,'USER AUTHORIZATION FAILURE');
              User_Authorization:= False;
              sound(220);
                end;
end;
               readln;
             end
       else
    begin
     OutTextXY(200,200,'USER AUTHORIZATION FAILURE');
    User_Authorization:= False; readln;CloseGraph;
    end;
    end;
    end;
    end.
```

```
unit testgr;
{*************************************************************************
  This unit is called by the main program TCG. MF. Cakmak and it is used to test
whether proper graphics hardware is already available in the computer.
*************************************************************************}
interface
uses Graph;
 GraphDriver,GraphMode,ErrorCode:Integer;

function Test_Graph_Device( GraphDriver:Integer;
                            ErrorCode:Integer;
                            GraphMode:Integer):Boolean;
implementation
 function Test_Graph_Device;
 begin
 GraphDriver:= Detect;
 InitGraph(GraphDriver,GraphMode,'');
 ErrorCode:= GraphResult;
  if ErrorCode <> grOk then
 begin
Test_Graph_Device:= False;
  Writeln('Program Aborted');
  Halt(1);
  end
  else
  Test_Graph_Device:= True;
  end;
  end.
```

```
unit perfile;
{*********************************************************************
  This unit is used for personel evaluations part of the main
  program. The unit is called by the outmenu.out procedure.
*********************************************************************}
   interface
   uses Crt,Dos;
   label 100;
   const
   X1 = 5; Y1 = 2; X2 = 75; Y2 = 22;
    zero = 0.0;
    type
    DateTime = record
    Year,Month,Day,Hour,Min,Sec:word
    end;
    astr = string[ 3 ];
    var
    Ch:Char; k:integer;
    Y3:integer;
    total,average:real;
    e:array (.1..10.) of real;
     Time:LongInt;
     DT:DateTime;
     Hour,Minute,Second,Sec100:word;
     Year,Month,Day,DayofWeek:word;
     lastname,name,filename,rank:string[ 12 ];
     textfile:text;
     str:astr;
     IOCode:integer;
answer : char;
procedure SetWindow(X1,Y1,X2,Y2:integer);
procedure HandleKey(Ch:Char);
procedure Condition(Low:integer ; var X: integer ; High : integer);
```

```
procedure SetXY(NewX,NewY : integer);
 procedure HandleFuncKey(Ch : Char );
 procedure main;
 implementation
    [************************************************]
    procedure SetWindow(X1,Y1,X2,Y2 :integer);
    const
    UpLeftCorner = #201 ;
    HorzBar      = #205 ;
    RightCorner = #187 ;
    VertBar      = #186 ;
    LowLeftCorner = #200 ;
    LowRightCorner = #188 ;
    Vert1Bar       = #179 ;
    Horz1Bar       = #196 ;
    var
    I,J : integer ;
    begin
    Window(X1-1,Y1-1,X2+1,Y2+1);
    ClrScr;
    Window(1,1,80,25);
    GotoXY(X1-1,Y1-1);
    Write(UpLeftCorner);
    for I:= X1 to X2 do
     Write(HorzBar);
    Write(UpRightCorner);
    for I:= Y1 to Y2 do
     begin
      GotoXY(X1-1,I);
 Write(VertBar);
 GotoXY(X2+1,I);
 Write(VertBar);
end;
GotoXY(X1-1,Y2+1);
Write(LowLeftCorner);
```

```
for I:=X1 to X2 do
 Write(HorzBar);
Write(LowRightCorner);
Window(X1,Y1,X2,Y2);
for I:=1 to 10 do begin
   for J:=X1 to X2 do
     Write(Horz1Bar);
   Y3:=WhereY+1;
   gotoXY(X1,Y3);
Write;
   end;
 for J:= Y1 to Y2 do
  begin
 GotoXY(2,J);
 write(VertBar);
 GotoXY(4,J);
 write(VertBar);
 GotoXY(3,J);
 Write(VertBar);
 GotoXY(1,J);
 write(VertBar);
 GotoXY(5,J);
 Write(Vert1Bar);
 GotoXY(25,J);
 Write(Vert1Bar);
 GotoXY(30,J);
write(Vert1Bar);
 GotoXY(60,J);
 Write(Vert1Bar);
 GotoXY(40,J);
 write(Vert1Bar);
end;
Window(X1,Y1,X2,Y2);
end;
procedure Handlekey(Ch:Char);
```

```
const
BEL = #7;
BS = #8;
CR = #13;
SP = #32;
 begin
 if Ch = BS then Write(BS,SP,BS)
 else if Ch = CR then Writeln
 else if Ch > = SP then Write(Ch)
 else if CH < > @Z then write(BEL)
  end;
procedure Condition(Low:integer;var X:integer ; High:Integer);
begin
if X < Low then X:= Low
else if X > High then X:= High;
end;
procedure SetXY(NewX,NewY:Integer);
begin
Condition(5,NewX,(1+X2-X1));
Condition(2,NewY,(1+Y2-Y1));
GotoXY(NewX,NewY);
end;
procedure HandleFuncKey(Ch:Char);
const
UpArrow = #72;
LeftArrow = #75;
RightArrow = #77;
DownArrow = #80;
begin
case Ch of
UpArrow : SetXY(WhereX,WhereY-1);
LeftArrow : SetXY(WhereX-1,WhereY);
RightArrow : SetXY(WhereX+1,WhereY);
DownArrow : SetXY(WhereX,WhereY+1);
end;
```

```
end;

(* procedure convert_to_real (str:astr; var r:real);
var i:integer;
begin
 case str of
 '0' : r:=0.0 ;   '6' : r:=6.0 ;
 '1' : r:=1.0 ;   '7' : r:=7.0 ;
 '2' : r:=2.0 ;   '8' : r:=8.0 ;
 '3' : r:=3.0 ;   '9' : r:=9.0 ;
 '4' : r:=4.0 ;
 '5' : r:=5.0 ;
 end
 else begin GotoXY(50,12);
  writeln('you should enter a number ');
 end;
 end; *)
 procedure main;
begin
[SI-]
 CheckBreak := False ;
 CheckEOF := False;
 DirectVideo := True ;
 repeat
   repeat
 total:=0;
 writeln('ENTER THE LAST NAME OF WHOM TO BE EVALUATED .. ');
 readln(lastname);
 ClrScr;
 writeln('ENTER THE FIRSTNAME OF WHOM TO BE EVALUATED .. ');
 readln(name);
 ClrScr;
 writeln('ENTER THE RANK OF WHOM TO BE EVALUATED .. ');
 readln(rank);
 ClrScr;
```

```
filename:=concat(lastname,'.',name);
writeln('FILENAME IS :',filename);
for k:= 1 to 13 do
writeln;
writeln('                E X E C U T I N G  . . . . . .');
for k:=1 to 15 do begin
 write('#');
 delay(250);
 end;
 ClrScr;
assign(textfile,filename);
reset(textfile);
  IOCode:=IOResult;
  if IOCode<>0 then begin
      writeln;
    writeln('FILE DOES NOT EXIST ');
      writeln;
      delay(500);
    write('DO YOU WANT TO CREATE A FILE ? ');
      delay(500);
       write('. ');
       delay(500);
       write(' . ');
      writeln('        Y / N ');
    readln(answer);
    if ((answer = 'Y') or (answer = 'y')) then begin
        if length(lastname)=2 then
            lastname:=concat(lastname,'      ')
        else if length(lastname)=3 then
            lastname:=concat(lastname,'     ')
        else if length(lastname)=4 then
            lastname :=concat(lastname,'    ')
        else if length(lastname)=5 then
            lastname :=concat(lastname,'   ')
        else if length(lastname)=6 then
```

```
            lastname := concat(lastname,'  ')
        else if length(lastname)= 7 then
            lastname := concat(lastname,' ')
        else writeln;
            filename:= concat(lastname,'.',name);
        writeln('THE FILE NAME CREATED IS : ',lastname);
        IOCode :=  IOResult;
        assign(textfile,filename);
        rewrite(textfile);
        for k:=  1 to 10 do begin
         writeln(textfile,total);
      end;
        close(textfile);
      end
      end;
      until IOCode=0;
      assign(textfile,filename);
      reset(textfile);
 while not EOF(textfile) do
  begin
   readln(textfile,lastname);
    for k:=1 to 10 do
     readln(textfile,e(.k.));
 end;
  close(textfile);
 if filename='' then  begin
    gotoXY(50,12);writeln('FILE NOT FOUND');
 end;
 SetWindow(X1,Y1,X2,Y2);
 gotoXY(5,2);
repeat
 gotoXY(42,2);
 writeln(lastname);
 for k:=1 to 10 do
 begin
```

```
gotoXY(32,2*k);
writeln(e(.k.):3:1);
end;
GOTOXY(8,2);
writeln('RELIABILITY');
GoTOXY(8,4);
writeln('RESPONSIBILITY');
GotoXY(8,6);
Writeln('LEADERSHIP');
GotoXY(7,8);
Writeln('UNDERSTANDABILITY');
GotoXY(10,10);
Writeln('STAMINA');
GotoXY(8,12);
Writeln('PERSONALITY');
GotoXY(8,14);
Writeln('GOOD MORAL ');
GotoXY(8,16);
Writeln('PRODUCTIVITY');
GotoXY(6,18);
Writeln('GENERAL BEHAVIOUR');
GotoXY(8,20);
writeln('PROFESSIONALITY');
GOTOXY(42,4);
Writeln(name);
GotoXY(42,6);
writeln(rank);
GotoXY(26,2);
Write(' ');
Readln(e(.1.);(* convert_to_real(str,e[ 1 ]);*)
GotoXY(26,4);
write(' ');
Readln(e(.2.));(* convert_to_real(str,e[ 2 ]);*)
GotoXY(26,6);
write(' ');
```

```
Readln(e(.3.)); (* convert_to_real(str,e[ 3 ]);*)
GotoXY(26,8);
write(' ');
Readln(e(.4.)); (* convert_to_real(str,e[ 4 ]):*)
GotoXY(26,10);
write(' ');
Readln(e(.5.));(* convert_to_real(str,e[ 5 ]);*)
GotoXY(26,12);
write(' ');
Readln(e(.6.)); (* convert_to_real(str,e[ 6 ]);*)
GotoXY(26,14);
write(' ');
Readln(e(.7.)); (* convert_to_real(str,e[ 7 ]);*)
GotoXY(26,16);
write(' ');
Readln(e(.8.)); (* convert_to_real(str.e[ 8 ]);*)
GotoXY(26,18);
write(' ');
Readln(e(.9.)); (* convert_to_real(str,e[ 9 ]);*)
GotoXY(26,20);Write(' ');
Readln(e(.10.)); (* convert_to_real(str.e[ 10 ]):*)
total:=0;
for k:= 1 to 10 do
total:=total+e(.k.);
average:=total 10;
GetDate(Year,Month,Day,DayofWeek);
GotoXY(50,14);      write('DATE');
GotoXY(62,14);
Write(Day,'_',Month,'_',Year);
GetTime(Hour,Minute,Second,Sec100);
GotoXY(50,16);
writeln('TIME');
gotoXY(62,16);
writeln(Hour,':',Minute,':',Second);
GotoXY(50,18); write('TOTAL');
```

```
GotoXY(66,18);write(total:3:1);
GotoXY(50,20);write('AVERAGE');
GotoXY(66,20);write(average:4:2);
GotoXY(41,2);Write(' ');
readln(lastname);
GotoXY(61,2);      writeln('SAVE ? "S" ');
GotoXY(61,4);      writeln('MODIFY 2xCR');
GOTOXY(61,6);      writeln('QUIT is              Z ');
READLN(Ch);
if (Ch = 's') or (Ch = 'S') then begin
assign(textfile,filename);
rewrite(textfile);
writeln(textfile,lastname,'          ');
for k:= 1 to 10 do
writeln(textfile,e[k]:3:1,' ');
close(textfile); end;
write(' '); readln(Ch);
Ch:= Readkey;
if Ch < > #0 then begin
   HandleKey(Ch);
   total:= 0;
end
elseHandleFuncKey(ReadKey);
until Ch =                                    Z;
Window(1,1,80,25);
ClrScr;
write('DO YOU WANT TO EVALUATE ANOTHER PERSON . . ? ');
writeln(' . Y / N ');
readln(answer);
until ((answer = 'n') or (answer = 'N'));
end;
end.
```

```
unit cic2:
[*************************************************************************]
    This unit is called  by the outmenu.out procedure if the user enters option 1 in the
main menu. CIC2 has 43 procedures and functions.
[*************************************************************************]
interface
uses Dos,Crt,timer,datefind,graph,testgr;
type
maxx = 1..80;
maxy = 1..25;
maxship = 1..30;
ownshiptype = record
  speed : real;
  course : real;
  latitude.longtitude : stringfff15";
  end;
targetship_at_t1 = record
  relative_speed: real;
  relative_bearing:real;
  true_bearing :real;
  speed : real;
  range : real;
  hour1.min1.sec1 :word;
  end;
targetship_at_t2 = record
  relative_speed : real;
  relative_bearing :real;
  true_bearing : real;
  range:real;
  speed:real;
  hour2,min2,sec2:word;
   end;
targetshiptype = record
      relative_speed : real;
      relative_bearing : real;
```

```
    speed : real;
    course : real;
    range : real;
    MCR : real;
    collision : boolean;
    air :boolean;
    dtime:longint;
     end;
  Targetship1_type = array (.maxship.) of targetship_at_t1;
  Targetship2_type = array (.maxship.) of targetship_at_t2;
  OwnShiptypes = ownshiptype;
  Targettype = array (.maxship.) of targetshiptype;
  var
  TargetShip1 : TargetShip1_type;
  TargetShip2 : TargetShip2_type;
  OwnShip : OwnShiptypes;
  Target : Targettype;
  sec100:word;
  shipcount1,shipcount2,shipcount,count:integer;
  nomore_ships : boolean;
  done,quit:boolean;
  continue,change_function :boolean;
  Ch1,Ch2,Ch3:char;
  x1 : maxx;
  y1 : maxy;
  totaltime:integer;
  which_ship_ch:char;

  [*  LIST OF PROCEDURE AND FUNCTIONS  *]
[*************************************************************]
  function deltatime(hour1,min1,sec1,hour2,min2,sec2 :word):longint;
  procedure timewrite(hourtemp,mintemp,sectemp : word ;
                      x : maxx ; y : maxy );
  procedure execution_signal(var xcoexe : maxx ; var ycoexe : maxy );
  procedure air_signal(var xcoair : maxx ; var ycoair : maxy);
```

```
procedure collision_signal(xcocol : maxx ;  ycocol : maxy ;
                  Target : Targettype ; shipcount : integer);
procedure Initialize_Screen;
procedure Execute(var Target1 : TargetShip1_type ;
             var Target2 : TargetShip2_type ;
                 MyShip  : OwnShiptypes;
             var Curr_Target : Targettype;
             var index   : integer );
function let_me_do_myship(which_ship_ch : char) : boolean;
procedure mywindow(var which_ship_ch:char; var quit : boolean);
procedure do_a(xin,yin,dx,dy:integer);
procedure do_b(xin,yin,dx,dy:integer);
procedure do_c(xin,yin,dx,dy:integer);
procedure do_d(xin,yin,dx,dy:integer);
procedure do_e(xin,yin,dx,dy:integer);
procedure do_f(xin,yin,dx,dy:integer);
procedure do_g(xin,yin,dx,dy:integer);
procedure do_one(xin,yin,dx,dy:integer);
procedure do_two(xin,yin,dx,dy:integer);
procedure do_three(xin,yin,dx,dy:integer);
procedure do_four(xin,yin,dx,dy:integer);
procedure do_five(xin,yin,dx,dy:integer);
procedure do_six(xin,yin,dx,dy:integer);
procedure do_seven(xin,yin,dx,dy:integer);
procedure do_eight(xin,yin,dx,dy:integer);
procedure do_nine(xin,yin,dx,dy:integer);
procedure do_zero(xin,yin,dx,dy:integer);
procedure Initialize(xin,yin,dx,dy:integer);
procedure Counts(x4,y4,dx,dy,counter : integer);
function Startgraphics : boolean ;
procedure radar_screen;
procedure sweep_screen1;
procedure sweep_screen3;
procedure sweep_screen4;
procedure sweep_screen2;
```

```
    procedure OwnShipplot;
    procedure TargetShipplot( var Target1 : TargetShip1_type;
                        var Target2 : TargetShip2_type;
                        var Target : Targettype;
                           shipcount : integer );
    procedure radar1(shipcount : integer); procedure do_myShip(var quit : boolean ; var
Change_function                 : boolean );
    function time_to_string(firsthour,firstmin,firstsec:word;
                    totaltime : integer ) : string;
    procedure do_guideship(var quit : boolean ; var change_function : boolean);
    procedure Thanks_for_Using_Cic;
    procedure main;
    [*** END OF THE LIST OF PROCEDURES AND FUNCTIONS ***]
    [*****************************************************************]
    implementation
 [ **********************     DELTATIME     ************************]
 [ This function  is used to convert the time difference between two given times into  the
value  of seconds  of which type longinteger.  In a given navigational problem, this
function  will accept any time difference  up to 68 year 18  days which is  a lot more than
what is needed for a specific navigational solution.

    Function  Deltatime is  a  real time  application, function  entries hour1 hour2,
min1 min2, sec1 sec2  are obtained via operating system and passed into the function
deltatime. Function Deltatime is called out  by the procedure do_OwnShip which is quite
a real_time application.  ]
     function deltatime(hour1,min1,sec1,hour2,min2,sec2 : word):longint;
 var
 timedifference :longint;
 begin
  if ((sec2  <  sec1) and (min2  <  min1)) then begin
        hour2: = hour2-1;
        min2: = min2 + 59;
        sec2: = sec2 + 60;
        end
   else if (( sec2  <  sec1 ) and ( min2  > =  min1)) then begin
         if min2 = min1 then begin
```

```
            hour2:= hour2-1;
            min2:= min2+59;
            sec2:= sec2+59;
            end
        else  begin
            min2:= min2-1;
            sec2:= sec2+60;
            end;
            end
   else if ((sec2 > = sec1) and (min2 < min1 )) then begin
        hour2:=(hour2)-1;
        min2 := min2 + 60;
        end
    else begin end;
      timedifference:=(((hour2-hour1)*3600) + ((min2-min1)*60) + (sec2-sec1));
      deltatime := timedifference;
      end; [function deltatime ]
[***************  END OF FUNTION DELTATIME  ******************]
[*********************        TIMEWRITE          ************************]
```

[This Procedure is used to write a given specific time into the specifed positions of the screen. The Specific time is designated by procedure parameters hourtemp, mintemp, and sectemp while the specific screen location is designated by the x,y coordinates.]

```
procedure timewrite(hourtemp,mintemp,sectemp : word ; x : maxx;
                                                      y : maxy );
begin
if ((mintemp < 10 ) and ( sectemp < 10 )) then begin
  gotoxy(x,y); write(hourtemp,':0',mintemp,':0',sectemp);
  end
 else if ((mintemp < 10) and (sectemp > = 10)) then begin
  gotoxy(x,y); write(hourtemp,':0',mintemp,':',sectemp);
  end
 else if ((mintemp > = 10) and (sectemp < 10)) then begin
  gotoxy(x,y); write(hourtemp,':',mintemp,':0',sectemp);
  end
```

```
 else begin
  gotoxy(x,y); write(hourtemp,':',mintemp,':',sectemp);
  end

  end;   [ procedure of procedure timewrite ]
```

[********** END OF THE PROCEDURE TIMEWRITE **********]

[********** EXECUTION SIGNAL **********]

[This procedure is used to signal that the execution is about to be done. Procedure parameters are designated screen coordinates.]

```
procedure execution_signal(var xcoexe:maxx; var ycoexe:maxy);
var
i : integer;
begin
for i := 1 to 3 do begin
  gotoxy(xcoexe,ycoexe);
  write('          ');
  delay(250);
  gotoxy(xcoexe,ycoexe);
  write('EXECUTION');
  delay(250);
  end;
  gotoxy(xcoexe,ycoexe); write('          ');
end; [ procedure execution_signal ]
```

[********** AIR SIGNAL **********]

This procedure will show any air target tracked in the designated screen location. The screen location is designated by the pair xcoair and ycoair.]

```
    procedure air_signal( var xcoair:maxx ; var ycoair:maxy );
var
 i : integer ;
```

```
begin
 for i := 1 to 4 do begin
  gotoxy(xcoair,ycoair);
  write('          ');
  delay(500);
  gotoxy(xcoair,ycoair);
  write(chr(177),chr(176),' AIR ',chr(176),chr(177));
  delay(500);
  end;
end; [ procedure air signal ]
```

[********** COLLISION SIGNAL **********]

This procedure will let the user know if the ownship course and speed is in conflict with any of the tracked targets' course and speed. If any of the targets will get as 500 yards (or lower) as close then the procedure warns the user as collision may happen by sounding in a special frequency and flashing in the corresponding screen location.

The procedure parameters are x,y screen coordinate pairs and Target with shipcount. This procedure is called by the do_Ownship procedure.]

```
procedure collision_signal(  xcocol : maxx ;
                             ycocol : maxy ;
                             Target : Targettype;
                             shipcount :integer);
 var
  i : integer ;
 begin
     for i := 1 to 3 do begin
      gotoxy(xcocol,ycocol);
      write(Target[shipcount].MCR :6:0);
      delay(1000);
      gotoxy(xcocol,ycocol);
      write('          ');
      Crt.Sound(560);
      delay(1000);
```

```
        Crt.NoSound;
        gotoxy(xcocol,ycocol);
        write('COLLISION');
        delay(500);
        gotoxy(xcocol,ycocol);
        write('          ');
        gotoxy(xcocol,ycocol);
        write(Target[shipcount].MCR:6:0);
        end;
        gotoxy(xcocol,ycocol);
        write(Target[shipcount].MCR:6:0);
end :[  end of procedure collision_signal ]
```

[********** END OF PROCEDURE COLLISION SIGNAL **********]

[********** INITIALIZE SCREEN **********]

This procedure does not have any parameters and is used to clean up data displayed in the corresponding screen locations. This will let the user enter his new data for new targets and new ownship and guideship navigational data. This procedure is called by do myship and go_guideship modules.]

```
procedure Initialize_Screen;
var
i:integer;
 begin
 timer.zaman(25,11,Ch); Ch := ReadKey;
  if Ch = #13 then begin
   i:= 13;
  repeat
     gotoxy(21,i);  write('                ');
        i:= i + 2;
        gotoxy(61,i);  write('                ');
   until  i  =  23;
       i:= 7;
       repeat
```

```
        gotoxy(61,i); write('                ');
        i:= i + 2;
        gotoxy(61,i);  write('                ');
        until i = 13;
        gotoxy(21,23); write('                ');
        gotoxy(61,23); write('                ');
  end;
end; [ end of procedure Initialize Screen ]

[ ***********************        EXECUTE        ************************** ]
[ ********************************************************************** ]
```

This procedure solves all the navigational equations, finds the results. The parameters are all data in the beginning and in the final positions of the targetships and OwnShip or the guideship and ownship. The total number of the ships is passed into the procedure. The output is Curr_Target which has all the data in record data structure of the what is needed. This procedure is called by the the modules do_myShip and do_guideship.]

```
    procedure execute(  var Target1:TargetShip1_type;
                var Target2: TargetShip2_type;
                        MyShip :OwnShiptypes ;
                        var Curr_Target : Targettype;
                        var index :integer );

const
one_radian = 57.2957;
one_mile = 2000;

var
    OwnShipx,OwnShipy,R1,R2,Teta,
    Teta1,Teta2,DeltaTeta,limit_angle_value:real;
    TargetShip1x,TargetShip2x,
    TargetShip1y,TargetShip2y,
    TargetShip3x,TargetShip4x,
```

```
    TargetShip3y,TargetShip4y,
    TargetShip5x,TargetShip5y,
    deltax,deltay,slope,
    inverse_slope,relative_motion,
    Relative_speed_in_yards,
    targetcourse_temp: real;
    i:longint;
sector1,sector2,sector3,sector4,
sector_north,sector_south,
 sector_east,sector_west : boolean ;
 begin
 sector1 := false;              sector2 := false;
 sector3 := false;              sector4 := false;
 sector_north:= false ;          sector_north:= false;
 sector_east:= false;            sector_west:= false;
    Curr_Target[ index ].collision := false;
     Curr_Target[ index ].air := false;
     if OwnShip.course = 0.0 then
        OwnShip.course:= OwnShip.course + 360.0;
     Teta := (OwnShip.course / one_radian);
    OwnShipx := 2000 * OwnShip.speed * sin( Teta );
    OwnShipy := 2000 * OwnShip.speed * cos( Teta );
    Teta1 := (OwnShip.course + Target1[ index ].relative_bearing) / ( one_radian);
    Teta2 := (OwnShip.course + Target2[ index ].relative_bearing) / ( one_radian);
    R1 := Target1[ index ].range;
    R2 := Target2[ index ].range;
    TargetShip1x := R1 * sin( Teta1 );
    TargetShip1y := R1 * cos( Teta1 );
    TargetShip2x := R2 * sin( Teta2 );
    TargetShip2y := R2 * cos( Teta2 );
     if ((TargetShip2x < > TargetShip1x) and (TargetShip2y = TargetShip1y)) then
        slope := 0
      else if ((TargetShip2x= TargetShip1x) and (TargetShip2y= TargetShip1y)) then
           slope := 0
      else if ((TargetShip2x= TargetShip1x) and (TargetShip2y< > TargetShip1y)) then
```

```
        slope := 1.6E7
    else slope := (TargetShip2y-TargetShip1y) (TargetShip2x-TargetShip1x);
    deltax := TargetShip1x - OwnShipx;
    deltay := TargetShip1y - OwnShipy;
    TargetShip3x := TargetShip2x - deltax;
    TargetShip3y := TargetShip2y - deltay;

    Relative_motion := sqrt(sqr(TargetShip2y - Targetship1y) +
                sqr(TargetShip2x - TargetShip1x));
    if Curr_Target[ index ].dtime < > 0 then
    [ this if statement will prevent crashing of the program if the user
hits enter accidentally ]
     Relative_speed_in_yards := (3600 * relative_motion) Curr_Target[ index ].dtime
    else Relative_speed_in_yards := (3600 * relative_motion) 1.0 ;
    if (Relative_motion < > 0 ) then begin
     TargetShip4x := ((((TargetShip3x-OwnShipx)*Relative_speed_in_yards) +
                (Relative_motion*OwnShipx)) (Relative_motion));

TargetShip4y := (OwnShipy - slope * (OwnShipx - TargetShip4x));
 if TargetShip4y < > 0 then
targetcourse_temp := Arctan(TargetShip4x / TargetShip4y)
 else  targetcourse_temp:=(1.6E6) (one_radian);
Curr_Target[index].course := targetcourse_temp * (one_radian);
i:= Round(Curr_Target[index].course);
i:= i mod 360;
Curr_Targetfflindex".course := i;
sector1 := ((TargetShip4x > 0) and (TargetShip4y > 0));
sector2 := ((TargetShip4x > 0) and (TargetShip4y < 0));
sector3 := ((TargetShip4x < 0 ) and (TargetShip4y < 0));
sector4 := ((TargetShip4x > 0) and (TargetShip4y < 0));
sector_north := ((TargetShip4x = 0) and ( TargetShip4y > 0));
sector_south := ((TargetShip4x = 0) and ( TargetShip4y < 0));
sector_east := ((TargetShip4x > 0 ) and ( TargetShip4y = 0));
sector_west := ((TargetShip4x < 0 ) and ( TargetShip4y = 0));
```

```
if sector2 then Curr_Target[ index ].course := Target[ index ]course-180
else if sector3 then
Curr_Target[index].course := Target[index].course + 180;
if sector_north then begin
 if (TargetShip4y < OwnShipy ) then
    Curr_Target[index].course := 180
    else if (TargetShip4y > OwnShipy) then
    Curr_Target[index].course := 0;
end
else if sector_south then begin
     if (TargetShip4y < OwnShipy) then
      Curr_Target[index].course := 180
     else if (TargetShip4y > OwnShipy) then
     Curr_Target[index].course := 0;
     end
else if sector_east then begin
     if (TargetShip4x < OwnShipx) then
     Curr_Target[index].course:= 270
     else if (TargetShip4x > OwnShipx) then
      Curr_Target[index].course := 90 ;
     end
     else if sector_west then begin
     if (TargetShip4x < OwnShipx) then
            Curr_Target[index].course := 270
     else if (TargetShip4x > OwnShipx ) then
            Curr_Target[index].course := 90;
     end;

 Curr_Target[index].speed := (sqrt(sqr(TargetShip4x)
                          + sqr(TargetShip4y)))/ one_mile;
end
else begin
Curr_Target[index].course := MyShip.course ;
Curr_Target[index].speed := MyShip.speed;
end;
```

```
if Curr_Target[index].course < 0.0 then
     Curr_Target[index].course := 360 + Curr_Target[index].course;


if slope = 0.0 then  inverse_slope := 1.6E7
else inverse_slope:= -1.0 slope;
TargetShip5x := (TargetShip1y - (slope * (TargetShip1x)))
                    / (inverse_slope - slope);
TargetShip5y := inverse_slope * TargetShip5x ;
DeltaTeta := abs( Teta2 - Teta1 );
if DeltaTeta > pi then DeltaTeta := (2*pi) - DeltaTeta;
if DeltaTeta < > 0.0 then begin
  if Relative_motion < > 0.0 then
     begin
    limit_angle_value := (sin(DeltaTeta) * Target2fflindex".range) /
                         ( Relative_motion );
     if ((limit_angle_value > = 1.0)) then
        Curr_Target[index].MCR := Target1[index].range
       else
      Curr_Target[index].MCR := sqrt(sqr(TargetShip5x) +
                                        sqr(TargetShip5y));
       if Target1[index].range < Target2[index].range then [ check again ]
        Curr_Target[index].MCR := Target1[index].range;
   end

 else Curr_Target[index].MCR := Target1[index].range;
end
else begin  [** DeltaTeta = 0 **]

  if Target1[index].range > TargetShip2[index].range then
     Curr_Target[index].MCR := 0.0
  else Curr_Target[index].MCR := Target1[index].range
end ;
Curr_Target[index].collision := Curr_Target[index].MCR < 500 ;
Curr_Target[index].air := Curr_Target[index].speed > 50;
```

```
end; [ * procedure execute * ]

    [************  END OF PROCEDURE EXECUTE  ************]
    [******************* LET_ME_DO_MYSHIP *******************]
```

[This function returns the boolean value true if User hits enter or Upper or lower case 'o'. The function parameter which_ship_ch is obtained and passed into the function.]

```
function  let_me_do_myship(which_ship_ch : char):boolean ;
var
 doit : boolean ;

begin
  doit := (which_ship_ch = #111) or (which_ship_ch = #79)
             or (which_ship_ch = #13);

  let_me_do_myship := doit;

  end; [ * function let_me_do_my_ship * ]

[**********************      MYWINDOW      **********************]
[ ***************************************************************]
```

This procedure draws the initial window of CIC program; OwnShip TargetShip, PivotShip data are always entered and the navigation problem solutions are displayed in this window. If there exist no navigational problem to be solved, window shows either the current time or the last solution presented by the software package. The current time representation and the data entries are managed by the automatic cursor movements, so that the user does not have to worry about which datum he/she should enter in which order. This procedure will pass two variables to the interfaced modules : One of them is which_ship_ch that is of type char, may be entered two show the intention of the user. The other variable passed to interfacing modules is that of quit which returns true if the user wants to quit program as soon as it runs. Procedure Mywindow is called by procedure do_myShip and procedure do_guideShip and main program as well.]

```
procedure mywindow (var  which_ship_ch :char ;
                        var  quit : boolean     );
var
i,k: integer;
begin
nomore_ships:= false;
done := false;

[     window(2,2,79,24);  ]
clrscr;
TextBackground(4);
TextColor(2);

window(1,1,80,25);
gotoxy(1,1);
write(#201);
for i:= 2 to 77 do
write(#205);
write(#187);
for i:= 2 to 23 do begin
 gotoxy(1,i);write(#186);
 gotoxy(78,i); write(#186);
 end;
 gotoxy(1,4); write(#204);
 gotoxy(1,24);
 write(#200);
 for i:= 2 to 77 do
  write(#205);
  write(#188);
  window(2,2,79,24);
   gotoxy(18,1); write(' N E V Z A T  G U L E S E N  P R E S E N T S ');
   gotoxy(23,2); writeln('C O M B A T  I N F O  C E N T E R ');
```

```
window(2,2,79,24);
gotoxy(1,3);
for i:=2 to 77 do
write(#205);
write(#185);
gotoxy(39,3);write(#203);
    for i:=4 to 22 do begin
     gotoxy(39,i); write(#186);
     end;
     gotoxy(39,23); write(#202);
     window(1,1,80,25);
     gotoxy(1,6);
     write(#204);
     for i:=2 to 39 do
      write(#205); write(#206);
     for i:= 41 to 77 do
      write(#205);
      write(#185);
      gotoxy(14,5);
      write('OWN SHIP DATA ');
      gotoxy(52,5);
      write('TARGET SHIP DATA ');
      for k:= 0 to 7 do begin
      gotoxy(1,8+(2*k));
      write(#199);
      for i:= 2 to 39 do
       write(#196);
       write(#215);
       for i:=41 to 77 do
       write(#196);
       write(#182);
      end;
      gotoxy(20,6);
      for i:= 7 to 23 do begin
      gotoxy(20,i);
```

```
write(#179);
gotoxy(60,i);
write(#179);
end;
gotoxy(20,6);
write(#209);
gotoxy(60,6);
write(#209);
gotoxy(20,24);
write(#207);
gotoxy(60,24);
write(#207);
 gotoxy(20,6);
for i:= 0 to 7 do begin
 gotoxy(20,8+(i*2));     write(#197);
 gotoxy(60,8+(i*2));     write(#197);
 end;
 gotoxy(3,7);
 write('T.C.G M.F.CAKMAK');
 gotoxy(24,7);
 write('STATUS . A . ');
 gotoxy(43,7);
 write('RELATIVE BEARING');
 gotoxy(7,9);
 write(' TODAY  ');
 datefind.tarih(21,9);
 gotoxy(48,9);
 write('RANGE');
 gotoxy(5,11);
 write('CURRENT TIME ');
 gotoxy(47,11);
 write('AT TIME');
 gotoxy(7,13);
 write('SPEED');
 gotoxy(43,13);
```

```
        write('RELATIVE BEARING');
        gotoxy(7,15);
        write('COURSE');
        gotoxy(48,15);
        write('RANGE');
        gotoxy(4,17);
        write('OWNSHIP / PIVOT');
        gotoxy(46,17);
        write('AT TIME');
        gotoxy(2,19);
        write(' CHANGE FUNCTION');
        gotoxy(43,19);
        write('TARGET SPEED');
        gotoxy(5,21);
        write('MORE SHIPS ?');
        gotoxy(43,21);
        write('TARGET COURSE  ',chr(233));
        gotoxy(4,23);
        write('TARGET_TRACKED ');
         gotoxy(48,23);
         write('MCR ');
         timer.zaman(25,11,Ch);
         Ch:= ReadKey;
         if Ch=#13 then begin end;
        gotoxy(26,17);
             Read(which_ship_ch);
             gotoxy(26,17);
             write(which_ship_ch);

        quit := (which_ship_ch =#113) or (which_ship_ch=#81);
 end;  [ procedure mywindow ]

[**************  END OF PROCEDURE MYWINDOW  **************]
[*********************************************************************]
[************* GROUP SEVEN SEGMENT PROCEDURES *************]
```

These seven segment procedures create seven segment display on the given screen coordinates.They draw lines to achieve the goal.The screen coordinates are given by the xin,yin parameters while the length and the width are assigned by the dx and dy parameters.This package may be employed to write script in future implementations as well. For the time being only the numbers from 0 to 9 has been implemented. The procedures form do_a through do_g are used to draw eigth approp_ riate lines. These group of procedures are used in graphics mode, so the package will require any of different kinds of graphic devices]

```
procedure do_a(xin,yin,dx,dy : integer);
 [ draws a line known as 'a' edge in seven segment display structure ]
 begin
   Line(xin,yin,xin + dx,yin);
 end;
procedure do_b(xin,yin,dx,dy : integer);
 [ draws a line known as 'b' edge in seven segment display structure ]
 begin
   Line(xin + dx,yin,xin + dx,yin + dy);
 end;
 procedure do_c(xin,yin,dx,dy : integer);
 [ draws a line known as 'c' edge in seven segment display structure ]
 begin
   Line(xin + dx,yin + dy,xin + dx,yin + (2*dy));
 end;
 procedure do_d(xin,yin,dx,dy :integer);
 [ draws a line known as'd' edge in seven segment display structure ]
 begin
  Line(xin,(yin + 2*(dy)),xin + dx,yin + 2*(dy));
 end;
 procedure do_e(xin,yin,dx,dy: integer);
 [ draws a line  known as 'e' edge in seven segment display structure ]
 begin
   Line(xin,yin,xin,yin + dy);
 end;
 procedure do_f(xin,yin,dx,dy : integer);
```

```
[ draws a line known as 'f' edge in seven segment display structure ]
begin
  Line(xin,yin + dy,xin,yin + 2*(dy));
end;
procedure do_g(xin,yin,dx,dy : integer);
[ draws a line known as 'g' edge in seven segment display structure ]
begin
 Line(xin,yin + dy,xin + dx,yin + dy);
end;
[********** END OF SEVEN SEGMENT GROUP PROCEDURES **********]
[*************************************************************************]
[******************  GROUP NUMBER PROCEDURES  ****************]
[ These procedures display numbers on the screen at the given screen coordinates. ]
   procedure do_one(xin,yin,dx,dy:integer);
 [ displays digital 1 in seven segment display structure ]
 begin
   do_b(xin,yin,dx,dy) :
     do_c(xin,yin,dx,dy);
 end;
procedure do_two(xin,yin,dx,dy:integer);
 [ displays digital 2 in seven segment display structure ]
 begin
    do_a(xin,yin,dx,dy);
      do_b(xin,yin,dx,dy);
    do_g(xin,yin,dx,dy);
      do_f(xin,yin,dx,dy);
      do_d(xin,yin,dx,dy);
 end;
procedure do_three(xin,yin,dx,dy:integer);
 [displays digital 3 in seven segment display structure ]
 begin
    do_a(xin,yin,dx,dy);
    do_b(xin,yin,dx,dy);
    do_c(xin,yin,dx,dy);
    do_g(xin,yin,dx,dy);
```

```
        do_d(xin,yin,dx,dy);
end;
procedure do_four(xin,yin,dx,dy:integer);
  [ displays digital 4 in seven segment display structure ]
 begin
     do_e(xin,yin,dx,dy);
     do_g(xin,yin,dx,dy);
     do_b(xin,yin,dx,dy);
        do_c(xin,yin,dx,dy);
 end;
procedure do_five(xin,yin,dx,dy:integer);
      [ displays digital 5 in seven segment display structure ]
 begin
    do_a(xin,yin,dx,dy);
    do_e(xin,yin,dx,dy);
    do_g(xin,yin,dx,dy);
       do_c(xin,yin,dx,dy);
       do_d(xin,yin,dx,dy);
 end;
procedure do_six(xin,yin,dx,dy:integer);
  [ displays digital 6 in seven segment display structure ]
 begin
    do_a(xin,yin,dx,dy);
    do_e(xin,yin,dx,dy);
    do_g(xin,yin,dx,dy);
       do_c(xin,yin,dx,dy);
    do_f(xin,yin,dx,dy);
       do_d(xin,yin,dx,dy);
end;
procedure do_seven(xin,yin,dx,dy:integer);
  [ displays digital 7 in seven segment display structure ]
 begin
    do_a(xin,yin,dx,dy);
    do_b(xin,yin,dx,dy);
    do_c(xin,yin,dx,dy);
```

```
 end;
procedure do_eight(xin,yin,dx,dy:integer);
    [ displays digital 8 in seven segment display structure . this procedure different from
the all others uses all the edges in seven segment display ]
      begin
    do_a(xin,yin,dx,dy);
       do_b(xin,yin,dx,dy);
    do_c(xin,yin,dx,dy);
       do_d(xin,yin,dx,dy);
    do_g(xin,yin,dx,dy);
       do_e(xin,yin,dx,dy);
       do_f(xin,yin,dx,dy);
 end;
procedure do_nine(xin,yin,dx,dy:integer);
[ displays digital 9 in seven segment display structure ]
 begin
    do_a(xin,yin,dx,dy);
       do_b(xin,yin,dx,dy);
    do_c(xin,yin,dx,dy);
       do_d(xin,yin,dx,dy);
    do_e(xin,yin,dx,dy);
       do_g(xin,yin,dx,dy);
 end;
procedure do_zero(xin,yin,dx,dy:integer);
[ displays digital zero in seven segment display structure ]
 begin
   do_a(xin,yin,dx,dy);
   do_b(xin,yin,dx,dy);
   do_c(xin,yin,dx,dy);
      do_d(xin,yin,dx,dy);
   do_e(xin,yin,dx,dy);
      do_f(xin,yin,dx,dy);
 end;
   [******** END OF THE GROUP NUMBER PROCEDURES  ********]
   [*************************************************************]
```

```
    procedure Initialize(xin,yin,dx,dy:integer);
    [This procedure is used to clear up the screen from previously displayed digital seven
segment numbers ]
    var i,k:integer;
     begin
 for i:= 0 to dx+1 do  begin
   OutTextXY(xin+i,yin,' ');
   for k:= 0 to (2*dy+1) do
     OutTextXY(xin,yin+i,' ');
 end;
end; [ procedure Initialize ]

procedure counts(x4,y4,dx,dy,counter:integer);
    [ This procedure is used to count (backward from nine to zero in this implementa-
tion ) using seven segment display.]

    var j:integer;
    begin

  for j := counter downto 0  do begin
   case j of
        9 :  begin  do_nine(x4,y4,dx,dy)  ;  delay(300); end;
        8 :  begin  do_eight(x4,y4,dx,dy) ;  delay(300); end;
        7 :  begin  do_seven(x4,y4,dx,dy) ;  delay(300); end;
        6 :  begin  do_six(x4,y4,dx,dy)   ;  delay(300); end;
        5 :  begin  do_five(x4,y4,dx,dy)  ;  delay(300); end;
        4 :  begin  do_four(x4,y4,dx,dy)  ;  delay(330); end;
        3 :  begin  do_three(x4,y4,dx,dy) ;  delay(330); end;
        2 :  begin  do_two(x4,y4,dx,dy)   ;  delay(330); end;
        1 :  begin  do_one(x4,y4,dx,dy)   ;  delay(330); end;
        0 :  begin  do_zero(x4,y4,dx,dy)  ;  delay(330); end;
        end; [ case statement ]
        ClearviewPort;
        end; [for loop ]
end; [ procedure ]
```

```
[********************** STARTGRAPHICS ************************]
function Startgraphics :boolean;
```

[This function checks the whether proper (or any) graphic device has been alreadyinstalled into the computer on which software package runs. The function will return true boolean value if the computer has the graphic devices.]

```
    var testgrok : boolean;
 begin
InitGraph(GraphDriver,GraphMode,' ');
testgrok := TestGr.Test_Graph_Device(GraphDriver,ErrorCode,GraphMode);
  if testgrok then   begin
   ClearViewPort;
   Startgraphics := testgrok;
   end
   else begin
      CloseGraph;
      Halt(1);
      end;
end; [function ]

procedure radar_screen;
```

[This procedure simulates the radar screen displaying whatever is seen in a real life radar]

```
    const
bigcirclescale = 50;
smallcirclescale = 2;
    var Xmid:integer;
   Ymid:integer;
   Asratio : real;
   i:integer;
   Palette : Palettetype;
begin
   ClearViewPort;
   Xmid:= GetMaxX div 2;
   Ymid:= GetMaxY div 2;
```

```
[ if Palette.size < > 1 then    (USE IF EGA GRAPHICS CARD  AVALILABLE) ]
 for i := 1 to 4 do begin
   Circle(GetMaxX div 2, GetMaxY div 2,bigcirclescale * i);
   Circle(GetMaxX div 2, GetMaxY div 2,smallcirclescale * i);
 end;

 Line(Xmid-210,Ymid,Xmid + 210,Ymid);
 Line(360,10,360,330);
              delay(100);

    OutTextXY(Xmid+215,171,'090');
    OutTextXY(Xmid-235,171,'270');
    OutTextXY(Xmid,10,'000');
    OutTextXY(Xmid,337,'180');
    OutTextXY(470,41,'030');
    OutTextXY(535,95,'060');
    OutTextXY(535,248,'120');
    OutTextXY(470,300,'150');
    OutTextXY(225,305,'210');
    OutTextXY(165,248,'240');
    OutTextXY(160,92,'300');
    OutTextXY(220,40,'330');

end; [ procedure radar_screen ]

[*************************************************************************]
[**********  SWEEP SCREEN GROUP PROCEDURES  *****************]
```

[There are four sweep screen procedures. These procedures sweep the radar screen to simulate a real life radar. They continuously draw lines from origin (0,0) of the coordinate system to the outer circle.

The procedures continuously calculate the coordinate of every single point which is on the outer circle. In calculation,procedure employs the general circle equation known as :

sqr(x-a)+sqr(y-b)=sqr(radius).]

```
procedure sweep_screen1;
  [ this procedure sweeps the first sector in the radar screen ]
var
Xmidd : integer;
Ymidd : integer;
AspRatio : real;
Xasp,Yasp : word;
j:integer;
xotemp:real;
begin
   GetAspectRatio(Xasp,Yasp);
   AspRatio:=Xasp Yasp;
   Xmidd := GetMaxX div 2;
   Ymidd := GetMaxY div 2;
   j:=-180;
   repeat
   xotemp:=Round(j);
      Line(Xmidd,Ymidd,Xmidd+j,Ymidd - Round(AspRatio *
                                          sqrt(abs(40000 - sqr(xotemp)))));
    delay(15);
    j:=j+1;
      until j=180;
end; [ procedure sweep_screen1  ]

procedure sweep_screen3;
 [ This procedure sweeps the third radar sector which is next to the
   first radar sector by implementation definition ]
var i : integer;
Xmid,Ymid : integer;
Asratio:real;
xtemp:real;
 Xasp,Yasp:word;
 begin
  i:=59;
  Xmid:=GetMaxX div 2;
```

```
  Ymid:= GetMaxY div 2;
  GetAspectRatio(Xasp,Yasp);
  Asratio := Xasp/Yasp;
  xtemp:= 0;
   repeat
     xtemp := i;
     Line(Xmid,Ymid,(Xmid + Round(200*sin(xtemp/52.295))),Ymid -
                          Round(AsRatio * (200*cos(xtemp/52.2957))));
     delay(25);
     i:= i + 1;
   until i= 120;
 end;[ procedure sweep_screen3 ]

 procedure sweep_screen4;
    [ This procedure sweeps the fourth radar sector which is next to the second radar
sector defined by implementation ]
       var
  i: integer;
  xtemp:real;
  Xmid,Ymid : integer;
  Asratio : real;
  Xasp,Yasp:word;
  begin
    Xmid := GetMaxX div 2;
    Ymid := GetMaxY div 2;
    GetAspectRatio(Xasp,Yasp);
    AsRatio := Xasp / Yasp;
    i:= 59;
    xtemp:= 0;
     repeat
       xtemp:= i;
       Line(Xmid,Ymid,Xmid-Round( 200 * sin(xtemp/52.295)),
                        Ymid+ Round(200 * AsRatio * cos(xtemp/52.295)));
       delay(25);
       i:= i + 1;
```

```
    until i = 122;
 end ; [ procedure sweep_screen4 ]

procedure sweep_screen2;
    [ This procedure will sweep the second radar sector which is before the fourth  and
after the third radar sector. This definition does not have any scientific importance and
is given by the author of the software package ]
     var i :integer;
   xtemp:real;
   Xmid,Ymid : integer;
   Xasp,Yasp:word;
   AsRatio : real;
begin
    Xmid := GetMAxX div 2; Ymid:= GetMaxY div 2;
      GetAspectRatio(Xasp,Yasp);
      Asratio := Xasp / Yasp;
      i:= -180;
      repeat;
        xtemp:= Round(i);
        Line(Xmid,Ymid,Xmid-i,Ymid + Round(AsRatio *
                                          sqrt(abs(40000 - sqr(xtemp)))));
        delay(15);
      i:= i + 1;
       until i = 180;
end; [procedure Sweep_screen2 ]

[********** END OF THE SWEEP SCREEN PROCEDURES **********]
[*****************************************************************]
[*************      PROCEDURE OWNSHIP PLOT      ***************]
```

This procedure plots the Ownship by employing ownship data rigth in the middle of the screen and radar display. In a real life radar it is not possible to see the OwnShip speed. This procedure also displays the own ship speed and course vector. In a real life radar it's also quite hard for the operator to see the ships' true course. This software package will show the true course and speed to the operator very clearly.]

```
     procedure OwnShipPlot;
```

```
const scale = 100;
var xtemp,ytemp,xadd,xsubt,yadd,ysubt : integer;
OwnShipx,OwnShipy:real;
Xasp, Yasp : word;
Xmid,Ymid :integer;
AsRatio : real;
begin
  Xmid := GetMaxX div 2;
  Ymid := GetMaxY div 2;
  GetAspectRatio(Xasp,Yasp);
  AsRatio := Xasp / Yasp;
  OwnShipx:= 2000*OwnShip.speed*sin(OwnShip.course / 57.2957);
  OwnShipy:= 2000*OwnShip.speed*cos(OwnShip.course / 57.2957);
  xtemp := abs(Round(OwnShipx / scale));
  ytemp := abs(Round(AsRatio * (OwnShipy / scale)));
  xadd := Xmid + xtemp ;          xsubt := Xmid - xtemp;
  yadd := Ymid + ytemp ;          ysubt := Ymid - ytemp;
  if Ownshipx > = 0 then xtemp := xadd
  else xtemp := xsubt;
  if (OwnShip.course > = 270) or (OwnShip.course < = 90) then
       ytemp:= ysubt
  else ytemp := yadd;
  Line(Xmid,Ymid,xtemp,ytemp);

end; [ procedure OwnShipPlot ]
[********** END OF THE PROCEDURE OWNSHIPPLOT **********]
[*****************************************************************]
[********** PROCEDURE  TARGETSHIP PLOT ******************]
```

[This procedure displays all the targets' speed and course vectors. The procedure also plots the target as a signal relative to the ownship position where is also relative to the middle of the radar and computer screen. In a real time radar, targets' courses and speeds are not displayed. This package solves the problem and makes it a lot easier to the radar operator. All the target data is passed into the procedure.]

```
procedure TargetShipplot(var Target1:TargetShip1_type;
                         var Target2:TargetShip2_type;
                         var Target:Targettype;
                         shipcount:integer);

const targetradius = 1;
      scale = 100;

var
    ArcCoords:ArcCoordsType;
    xtemp,ytemp,i,k,xtemp2,ytemp2 :longint;
    xadd,yadd,xsubt,ysubt : integer;
    TargetShip1x,TargetShip1y:array (.maxship.) of real;
    TargetShip6x,TargetShip6y:array (.maxship.) of longint;
    xtemp1,AsRatio:real;
    deltax,deltay:longint;
    ship_id :string;
    Xasp,Yasp:word;
    Xmid,Ymid : integer;
begin
GetAspectRatio(Xasp,Yasp);
AsRatio:=(Xasp Yasp);
Xmid := GetMaxX div 2;  Ymid := GetMaxY div 2;

for k:= 1 to shipcount do begin
  TargetShip1x[k] := Target1[k].range * sin((OwnShip.course +
                     Target1[k].relative_bearing)/57.2957);
  TargetShip1y[k] := Target1[k].range * cos((OwnShip.course +
                     Target1[k].relative_bearing)/57.2957);
   TargetShip6x[k]:= Round(Target[k].speed * 2000 *
                          Sin(Target[k].course/57.2957)/(scale));
  TargetShip6y[k]:= Round(Target[k].speed *2000 *
                         AsRatio * (Cos(Target[k].course/57.2957)/(scale)));
  deltax := TargetShip6x[k]; deltay:= TargetShip6y[k];
  xtemp1 := (TargetShip1x[k] scale);
```

```
xtemp:=abs(round(xtemp1));
ytemp := (Round( AsRatio * (TargetShip1y[k] scale)));
xadd := xtemp + Xmid;
xsubt := Xmid - xtemp ;
   if TargetShip1x[k] > 0 then xtemp := xadd
   else xtemp := xsubt;
   if ytemp > 0 then ytemp := Ymid - ytemp
   else ytemp := abs(ytemp) + Ymid;
    for i := 1 to 4 do
    Circle(xtemp,ytemp,targetradius*3); [ draw the signal on the radar ]
    xtemp2:=xtemp + deltax;
  if deltay > = 0 then
     ytemp2 := ytemp - deltay
     else ytemp2 := ytemp + abs(deltay);
[************************************************************** ]
[**   Arc Solution to the unknown Targetship coordinates as in
[**   real life radar solution
[**  if Target[ k ].Course < = 90 then
[**  Arc(xtemp,ytemp,90,90 - Round(Target[k].course),
          Round(Target[k].speed  * 2000 / scale ))
           else
           Arc(xtemp,ytemp,90,450 - Round(Target[k].course),
           Round(Target[k].speed  * 2000 / scale ));
          GetArcCoords(ArcCoords);
     with ArcCoords do
[ ** [Xstart := xtemp; Ystart:=ytemp;
[ **  Line(xtemp,ytemp,Xend,Yend);
[*************************************************************]

           Line(xtemp,ytemp,xtemp2,ytemp2);
           if ((xtemp2 > GetMaxX) or (xtemp2 < 0) or (ytemp2 > GetMaxY)
          or (ytemp2 < 0)) then
           OutTextXY(xtemp+10,ytemp-10,chr(11));
          if k < 10 then  begin
          Ship_id := chr(48+k);
```

```
            OutTextXY(xtemp+5,ytemp,Ship_id);
          end;
      end;
end; [procedure targetshipplot ]
[*************** END OF THE TARGETSHIPPLOT ********************]

[******************** PROCEDURE RADAR1 ************************]

procedure radar1(shipcount : integer);
   [this procedure handles all the radar operations calling the group procedures.]
    var
k,kten,total : integer;
xtemp : integer;
Xstart,Ystart,delx,dely : integer;
testgrOk:boolean;
AsRatio : real;
Xmid,Ymid : integer;
xin,yin,dx,dy:integer;
Xasp,Yasp:word;
 begin [radar1]
  if Startgraphics then begin   [ check out whether the proper
                          graphic device is already installed ]
[** This following code segment will present author name and stand-by the
                  user for the radar screen                    ** ]
    SetTextStyle(1,Horizdir,1);
    OutTextXY(140,150,'N E V Z A T  G U L E S E N  P R E S E N T S');
    Line(130,173,585,173);
    delay(1000);
    ClearviewPort;
    SetTextstyle(0,Horizdir,0);
    Sound(500);
    delay(120);
    NoSound;
    radar_screen; delay(1000);
    ClearViewPort;
```

```
    Xmid := GetMaxX div 2;
      Ymid := GetMaxY div 2;
    xin := xmid + 240;
      yin := ymid - 130;
      dx:= 50;     dy:= 35;
    counts(xin,yin,dx,dy,9);
    k:= 0;
    radar_screen;
   repeat
    xin := Xmid + 240;
      yin := ymid - 130;
      dx := 25;     dy := 15;
[ following two case statements will show the number of the full rotations
 of the radar has already done ]
    case kten of
     1: do_one(xin,yin,dx,dy);  2:do_two(xin,yin,dx,dy);
     3:do_three(xin,yin,dx,dy); 4:do_four(xin,yin,dx,dy);
     5:do_five(xin,yin,dx,dy);  6:do_six(xin,yin,dx,dy);
     7:do_seven(xin,yin,dx,dy); 8:do_eight(xin,yin,dx,dy);
     9:do_nine(xin,yin,dx,dy);  0:do_zero(xin,yin,dx,dy);
     end;
     delay(500);
     xin := xmid - 240;
       yin := ymid - 130;
       dx := 25; dy := 15;
     case total of
     1: do_one(xin,yin,dx,dy) ; 2: do_two(xin,yin,dx,dy);
     3:do_three(xin,yin,dx,dy); 4:do_four(xin,yin,dx,dy);
     5: do_five(xin,yin,dx,dy); 6: do_six(xin,yin,dx,dy);
     7:do_seven(xin,yin,dx,dy); 8:do_eight(xin,yin,dx,dy);
     9:do_nine(xin,yin,dx,dy);  0:do_zero(xin,yin,dx,dy);
     end;
     delay(500);
     Rectangle(1,1,GetMaxX-1,GetMaxY-1);
     radar_screen;
```

```
      OwnShipPlot;
      if k > = 1 then begin
         Rectangle(1,1,GetMaxX-1,GetMaxY-1);
          TargetShipplot(TargetShip1,TargetShip2,Target,shipcount);
          delay(100);
      end;
[ ** start sweeping radar screen , as in real life radar ]
      sweep_screen1;
        sweep_screen3;
      radar_screen; [ show radar screen back ]
      OwnShipPlot;  [ plot your own ship employing data in procedure execute ]
      Rectangle(1,1,GetMaxX-1,GetMaxY-1);
[ plot the Target employing data from procedure execute ]
     TargetShipPlot(targetShip1,TargetShip2,Target,Shipcount);
     delay(100);
[ sweep the secondary half of the radar screen ]
     sweep_screen2;
       sweep_screen4;
     OwnShipplot;   [ plot your own ship back in ]
     if k > 1 then  begin
        RecTangle(1,1,GetMaxX-1,GetMaxY-1);
     TargetShipPlot(TargetShip1,TargetShip2,Target,shipcount);
     delay(100);
   end;

     k:=k+1; [ increment the value of full rotations of the radar ]
     kten := k mod 10;
     if k < 100 then
        total := k div 10
     else if (k> = 100) and (k < 1000) then
        total:= k div 100
     else total := k div 1000;
  until KeyPressed;
end         [if graph device is ok ]
else begin  [ if there is no graphic device in the working computer
```

```
                    program will halt ]
      Closegraph;
        Halt(1);
  end;  [ else ]
  CloseGraph;   [ Task is done , go back to Text mode ]
  end; [radar1 ]
  [************** END OF PROCEDURE RADAR1  ********************]
  [*************************************************************************]
  [************** END OF ALL THE RADAR PACKAGES **************]
  [************************     DO_MYSHIP     ************************]
  [*************************************************************************]
```

This is one of the main module in the package. Procedure do_myship handles all the calculations and solutions calling procedure execute if the user wants to track other targets.]

```
      procedure do_myship( var quit:boolean;
                           var change_function:boolean );

  var
   i: integer;
   Ch,Ch1,Ch2,Ch3,Ch4:char;
   see_radar,no_more_myShip : boolean;
  begin
       quit := false;
         see_radar:= false;
         no_more_myShip:= false;
         change_function:= false;
[*200*]  repeat
         timer.zaman(25,11,Ch) ; Ch := ReadKey;
             if Ch = #13 then begin
               gotoxy(25,13);
               Read(OwnShip.speed);
               gotoxy(29,13); write('knots');
               gotoxy(27,15);
               Read(OwnShip.course);
               gotoxy(30,15); write(chr(248));
```

```
        end;
        ShipCount1 := 0;
        [** START GETTING FIRST POSITION OF TARGET **]
[50]    repeat
        shipcount1 := shipcount1 + 1;
      timer.zaman(25,11,Ch);
      Ch:= ReadKey;
        if Ch = #13 then begin
         gotoxy(25,23);
         write( shipcount1:4);
         gotoxy(68,7);
         read(TargetShip1[shipcount1].relative_bearing);
         gotoxy(71,7); write(chr(248));
         gotoxy(67,9);
         read(TargetShip1[shipcount1].range);
        end;
        timer.zaman(25,11,Ch);
        Ch:= ReadKey;
        if Ch=#13 then begin
          TargetShip1[shipcount1].min1:= Min;
          TargetShip1[shipcount1].Hour1:= Hour;
          TargetShip1[shipcount1].sec1:= Sec;
      timewrite(TargetShip1[shipcount1].hour1,
                        TargetShip1[shipcount1].min1,
                          TargetShip1[shipcount1].sec1,65,11);
         gotoxy(27,21); read(Ch1);
         Ch1:= ReadKey;
         gotoxy(27,21); write(Ch1);
         nomore_ships := ((Ch1=#110) or (Ch1=#78));
         end;
         if not(nomore_ships) then begin
              gotoxy(68,7);write('        ');
              gotoxy(67,9);write('        ');
              gotoxy(65,11); write('          ');
              end;
```

```
            until nomore_ships;     [** LOOP TO 50 **]

            [ ** START READING SECONDARY POSITION OF THE TARGETS
*]
            shipcount2 := 0;
  [100]     repeat
            shipcount2 := shipcount2 + 1;
            gotoxy(25,23);
            write('      ');
            gotoxy(25,23);
            write(shipcount2);
            timer.zaman(25,11,Ch);
            Ch:= ReadKey;
            if Ch=#13 then begin end ;
            GetTime(Hour,min,sec,Sec100);
            TargetShip2[shipcount2].hour2:= Hour;
            TargetShip2[shipcount2].min2 := min ;
            TargetShip2[shipcount2].sec2 := sec;

            gotoxy(68,13);
            read(TargetShip2[shipcount2].relative_bearing);
            gotoxy(71,13); write(chr(248));
            gotoxy(67,15);
            read(TargetShip2[shipcount2].range);
            timewrite(TargetShip2[shipcount2].hour2,
                    TargetShip2[shipcount2].min2,
                    TargetShip2[shipcount2].sec2,65,17);
            delay(100);
          if shipcount2 < shipcount1 then begin
                  gotoxy(68,13);
                  write('      ');
                  gotoxy(67,15);
                  write('      ');
                  gotoxy(65,17);
                  write('      ');
```

```
                end;
            until (shipcount1 = shipcount2); [** LOOP TO 100 **]

  [************************ EXECUTION ********************************]
    [***] for i := 1 to shipcount2 do                          [***]
    [***] [ * find timedifference for each target *]           [***]
            Target[i].dtime :=
                deltatime(TargetShip1[i].hour1,TargetShip1[i].min1,
           TargetShip1[i].sec1, TargetShip2[i].hour2,TargetShip2[i].min2,
       TargetShip2[i].sec2);
            x1:=65 ; y1 := 2;
            timer.zaman(25,11,Ch);
            if Ch=#13 then
              execution_signal(x1,y1);
            [* if return hit then start calculation for each
                signaling that the computer executing *]
            for count:= 1 to shipcount2 do
       execute(TargetShip1,TargetShip2,OwnShip,Target,count);
             Continue := true;
    [***]   done:=false;                                      [***]
    [***]   shipcount:=0;                                     [***]
             [ Ch2:=#77; ]
  [*******************************************************************]
[150]  repeat
            window(1,1,80,25);
         gotoxy(1,22);
            write(#199);
            done:=false;
        shipcount := shipcount + 1;

          timer.zaman(25,11,Ch);
            Ch:=ReadKey;
            if Ch = #13 then begin end;
             gotoxy(25,23);
             write('      ');
```

```
        gotoxy(27,23);
        write(shipcount);
        gotoxy(67,21);
        write('       ');
        gotoxy(27,21);
        write('       ');

        gotoxy(61,19);
        write(Target[shipcount].speed:8:2,'knots');
        if ((Target[shipcount].course < 100) and
            (Target[shipcount].course > = 10)) then begin
            gotoxy(67,21);
            write('0',Target[shipcount].course:2:0,chr(248));
         end
        else if (Target[shipcount].course < 10 ) then
            begin
           gotoxy(67,21);
           write('00',Target[ shipcount ].course:1:0,chr(248))
       ;end
        else
            begin
         gotoxy(67,21);
write(Target[shipcount].course:3:0,chr(248));
         end;
         gotoxy(63,23);
         write('         ');
         gotoxy(63,23);
         write(Target[shipcount].MCR :6:0);
          if Target[shipcount].air then
              begin
             x1 := 28;       y1 := 23;
       air_signal(x1,y1);
                gotoxy(21,23);write(shipcount:2);
      end;
            gotoxy(21,23);
```

```
            write(shipcount:2);
            if not(Target[shipcount].air) then
              begin
               gotoxy(28,23);
               write('          ');
               end;
          if Target[shipcount].collision then
             begin
               x1 := 63; y1 := 23;
        collision_signal(x1,y1,Target,shipcount);
             end;

          if shipcount < shipcount1 then begin
          for i:= 1 to 2 do begin
          gotoxy(5,21);
          write('          ');
          delay(250);
          gotoxy(5,21);
          write('NEXT TARGET');
          delay(250);
          end;
          gotoxy(27,21);
          read(Ch2);
          Ch2:= ReadKey;
          Continue:=(Ch2=#13) or (Ch2=#89) or (Ch2=#121);
          end
          else begin
            done := true;
      gotoxy(1,22); write(#199);
          end
until ((done) or (not(Continue)));  [* LOOP TO 150 *]
          gotoxy(5,21);
          write('MORE SHIPS ');
          gotoxy(27,21);
          Read(Ch3);
```

```
            Ch3:= ReadKey;
            No_more_myship:= (Ch3= #78) or (Ch3= #110);
            if No_more_myShip= true then
                begin
                gotoxy(28,21); write(Ch3);
                gotoxy(28,19);
                  Read(Ch4); Ch4:= ReadKey;
                  Change_function:= (Ch4= #89) or (Ch4= #121);  [y]
          quit :=  (Ch4= #81) or (Ch4= #113);          [q]
           see_radar:= (Ch4= #114) or (Ch= #82);
              end
       else begin Change_function :=  false;
                        quit :=  false;
                  end;
            if see_radar then
              radar1(shipCount);
        Initialize_Screen;
         until (Change_function) or  (no_more_Myship) or quit;
            [* LOOP TO 200 *]
            mywindow(which_ship_ch,quit);
end; [ procedure do_ownship ]

[*************  END OF PROCEDURE DO_OWNSHIP  *************]

[*************** FUNCTION TIME TO STRING ********************]
[************************************************************]
   [ This function gets the starting time and the given time difference and returns the
string value that represents the total value of the starting time and the timedifference
]
   function time_to_string(firsthour,firstmin,firstsec:word;
            totaltime : integer) : string;

 var
 lasthour,lastmin,lastsec,
 temphour,tempmin,tempsec:integer ;
```

```
firstinthour,firstintmin,firstintsec : integer;
laststrhour,laststrmin,laststrsec : string[2];
laststring:string[8];
see_radar:boolean;

begin
firstinthour := firsthour;
firstintmin := firstmin;
firstintsec := firstsec; [* assignment statements for type compatibility *]
[ * start applying what software methodology course taugth us *]
  temphour := totaltime div 3600;
  totaltime := totaltime mod 3600; [ update totaltime difference ]
  tempmin := totaltime div 60;
  tempsec := totaltime mod 60;
  lasthour := temphour + firstinthour;
  lastmin := tempmin + firstintmin;
  lastsec := tempsec + firstintsec;
  if lastsec > 59 then begin
       lastsec := lastsec - 60;
       lastmin := lastmin + 1;
  end;
  if lastmin > 59 then begin
       lastmin := lastmin - 60;
       lasthour := lasthour + 1;
  end;
  if lasthour > 23 then lasthour := lasthour mod 24;
  Str(lastsec,laststrsec);
     if length(laststrsec) < 2 then
        laststrsec:= concat('0',laststrsec);
  Str(lastmin,laststrmin);
     if length(laststrmin) < 2 then
        laststrmin:= concat('0',laststrmin);
  Str(lasthour,laststrhour);
      if length(laststrhour) < 2 then
        laststrhour:= concat('0',laststrhour);
```

```
   laststring:= concat(laststrhour,':',laststrmin,':',laststrsec);
   time_to_string := laststring;

end; [ procedure time_to_string ]
[************** END OF PROCEDURE TIME_TO_STRING *************]

[************************ DO_GUIDESHIP ***************************]
[*****************************************************************]
```

[This is also one of the main procedure of the software package. This procedure is called by the main program and executes as long as the user wants to do navigational calculations to figure out what his/her own ship data should be in order to get some specific location in a given time where is relative to the commanding ship. The procedure parameters are quit which tells the computer that the user wants to quit, and change_function which directs the screen data entry location to do another type of solutions of which software package has been implemented for.]

```
      procedure do_guideship(var quit: boolean ; var change_function : boolean);
 var NextChar,Ch1,Ch2,Ch3 : char ;
  see_radar, Nomore_pivotShip : boolean;
 begin     [ do pivot ship ]
  quit:= false;
  Change_function:= false;
  see_radar:= false;
  Nomore_pivotship:= false;
  repeat
  window(1,1,80,25);
  gotoxy(14,5); write('     ');     [ Start with changing window header ]
  gotoxy(14,5); write('PIVOT SHIP DATA ');
  gotoxy(52,5); write('     ');
  gotoxy(53,5); write(' OWN SHIP DATA   ');
  gotoxy(43,19); write('OWNSHIP SPEED ');
  gotoxy(43,21); write('OWNSHIP COURSE');
  [    repeat                    ]
       window(1,1,80,25);
       gotoxy(25,13);
       Read(OwnShip.speed);
```

```
gotoxy(29,13);
write('knots');
gotoxy(27,15);
Read(OwnShip.course);
gotoxy(30,15) ;
write(chr(248));
 timer.zaman(25,11,Ch);
 Ch:= ReadKey;
  if Ch=#13 then begin
       gotoxy(68,7);
          Read(TargetShip1[1].relative_bearing);
       gotoxy(71,7);
          write(chr(248));
          delay(100);
       gotoxy(67,9);
          Read(TargetShip1[1].range);

        TargetShip1[1].min1 := Min;
        TargetShip1[1].hour1 := Hour;
        TargetShip1[1].sec1 := Sec;
      end;
    [ myship became target in solution of the problem ]
   timewrite(TargetShip1[1].hour1,TargetShip1[1].min1,
                TargetShip1[1].sec1,65,11);
timer.zaman(25,11,Ch);          Ch:= ReadKey;
   if Ch = #13 then begin
      gotoxy(68,13);
      Read(TargetShip2[1].relative_bearing);
   gotoxy(71,13);
        write(chr(248));
        delay(100);
      gotoxy(68,15);
        Read(TargetShip2fff1".range);
      gotoxy(4,23);  write('        ');
      gotoxy(4,23);  write(chr(228),'t');
```

```
      gotoxy(24,23); write('          ');

      gotoxy(24,23); Read(totaltime);
      gotoxy(30,23); write('seconds');
    end;
     Target[1].dtime := totaltime;
     timer.zaman(25,11,Ch); Ch := ReadKey;
     if Ch = #13 then
         begin
          x1 := 65; y1 := 2;
          execution_signal(x1,y1);
          count := 1;
          execute(TargetShip1,TargetShip2,OwnShip,Target,count);
      end;
   gotoxy(64,17);
   write(time_to_string(TargetShip1[1].hour1,
                        TargetShip1[1].min1,TargetShip1[1].sec1,totaltime));
gotoxy(61,19);
   write(Target[1].speed:8:2,'knots');
   if ((Target[1].course < 100) and (Target[1].course > = 10)) then
   begin
        gotoxy(67,21);
        write('0',Target[1].course:2:0,chr(248));
   end
   else if (Target[1].course < 10) then
       begin
         gotoxy(67,21);
         write('00',Target[1].course:1:0,chr(248));
   end
   else begin
         gotoxy(67,21);
         write(Target[1].course:3:0,chr(248));
         end;
 gotoxy(63,23);
    write('          ');
```

```
    gotoxy(63,23);
    write(Target[1].MCR:6:0);
[ if solution tends to collision then alarm ..! , given relative
     position is wrong .. !, need better C.O.'s ]
  [*    if collision case wanted to be implemented in this type of solution
        then the following code segment will be taken out of the comment
        marks *]
  [*************************************************]
         if Target[ 1 ].collision then  begin
         x1 := 63; y1:= 23;
         count:= 1;
         collision_signal(x1,y1,Target.count);
         end;
  [*************************************************]

        timer.zaman(25,11,Ch);  Ch:= ReadKey;
        if Ch=#13 then begin   end;
         gotoxy(27,21);  Read(NextChar);
         NextChar:= ReadKey;
         nomore_pivotship := ((NextChar = #113) or (NextChar = #81) or
                    (NextChar = #110) or (NextChar = #78)); [q, n]

         if nomore_pivotship= false then
             begin
              change_function:= false;
              quit:= false;
              end
         else
             begin
         gotoxy(28,19);
         Read(Ch3);       Ch3:= ReadKey;
         Change_function :=  (Ch=#89) or (Ch=#121);
         quit :=  ((Ch3=#81) or (Ch3=#113));
         see_radar:= ((Ch3=#82) or (Ch3=#114));
         if see_radar then
```

```
            radar1(1); Initialize_SCreen;
                end;

            until quit or change_function or nomore_pivotship;
        mywindow(which_ship_ch,quit);


end; [ end of pivot ship positioning solution ]

[******** END OF PROCEDURE DO_GUIDESHIP **********]

[******** PROCEDURE THANKS FOR USING CIC **********]
This procedure will print out a 'good_bye'message to the user as the user exits the program. Procedure does not have any parameter and only uses Crt and Operating system timing routines. It's an attempt to make computers more sympatic to whom use it.]
    procedure Thanks_for_Using_Cic;
 var
  i: integer ;
 begin
  for i:= 1 to 2 do begin
       ClrScr;
       Crt.Sound(50);
       delay(250);
       Crt.NoSound;
       gotoxy(18,12);
       write('T H A N K  Y O U  F O R  U S I N G  C. I. C. ');
       delay(1000);
      end;
end;  [ procedure thank_for_using_cic ]

[********** END OF PROCEDURE THANKS FOR USING CIC **********]
[*************************************************************]
 procedure main ;

 begin      [ ***** main ****** ]
```

```
    quit:= false;
    TextBackground(4);
          repeat
          mywindow(which_ship_ch,quit);
           while not(quit) do begin
               if Let_me_do_myship(which_ship_ch) then begin
                     do_myship(quit,change_function);
  if change_function then do_guideship(quit,change_function)
                      end
          else begin
              do_guideship(quit,change_function);
              if change_function then do_myship(quit,change_function);
                    end;
           end;  [while]
           until quit;
           Thanks_for_using_cic;
           end;
end.
```

```
unit Datefind;
interface
    uses Crt,Dos;
    type
     xcoord = 1..80;
ycoord = 1..25;
DateTime = record
Year,Month,Day:word;
 end;
var
daystring:string[ 9 ];
DT:DateTime;
Year,Month,Day,DayOfWeek : word;
procedure tarih(x:xcoord ; y:ycoord);
implementation
procedure tarih(x:xcoord ; y:ycoord);
begin
GetDate(Year,Month,Day,DayOfWeek);
 case DayofWeek of
  1: daystring := 'MONDAY' ;
  2: daystring := 'TUESDAY';
  3: daystring := 'WEDNESDAY';
  4: daystring := 'THURSDAY';
  5: daystring := 'FRIDAY';
  6: daystring := 'SATURDAY';
  0: daystring := 'SUNDAY';
  end;
  window(1,1,80,25);
 gotoxy(x,y);
 write(daystring,' ',Month,'/',Day,'/',Year);
 end;
 end.
```

```
Unit timer;
interface
Uses Dos,Crt;
type
 DateTime = record
   Year,Month,Day,Hour,Min,Sec : word;
   end;
   xcoord= 1..80;
   ycoord= 1..25;
var
Ch:char;
Time : LongInt;
DT : DateTime;
Hour,Min,Sec,Sec100 :word;
Year,Month,Day,DayOfWeek:word;
procedure zaman(x:xcoord;y:ycoord ; var Ch:char);
implementation
procedure zaman(x:xcoord;y:ycoord ; var Ch:char);
begin
gotoxy(x,y);
repeat
 GetTime(Hour,Min,Sec,Sec100);
 gotoxy(x,y);
 if ((Sec  <  10 ) and (Min  > = 10 )) then
 write(Hour,':',Min,':0',Sec,':',Sec100)
 else if ((Sec  > =  10 ) and (Min  <  10 )) then
 write(Hour,':0',Min,':',Sec,':',Sec100)
 else if ((Sec  <  10 ) and ( Min  <  10 )) then
 write(Hour,':0',Min,':0',Sec,':',Sec100)
 else  write(Hour,':',Min,':',Sec,':',Sec100);
 until KeyPressed;
 end;
 end.
```

```
unit dm1;
[* This unit is called by unit dm8. It displays the ship  general
 information  and draws the ship in graphics mode of the screen *]
interface
uses Graph,Dos,TestGr;
const
CakmakShip : array [ 1..7 ] of PointType = (( x:45 ; y:100),
                                ( x:360 ; y:100),
                                ( x:675 ; y:90),
                                ( x:660 ; y:150),
                                ( x:215 ; y:150),
                                ( x:50 ;y:130),
                                ( x:45 ;y:100));

procedure main;
implementation
procedure main;
begin
if not Test_Graph_Device(GraphDriver,ErrorCode,GraphMode) then
 begin
  writeln('Program aborted');
  Halt(1);
end
else
begin
   Line(657,92,657,150);
   Line(630,93,630,150);
   Line(585,94,585,150);
   Line(540,95,540,150);
   Line(510,96,510,150);
   Line(480,97,480,150);
   Line(420,98,420,150);
   Line(360,100,360,150);
   Line(345,100,345,150);
   Line(285,100,285,150);
```

```
    Line(215,100,215,150);
    Line(180,100,180,145);
    Line(150,100,150,142);
    Line(115,100,115,137);
    Line(75,100,75,133);
    Line(60,100,60,131);
    Line(215,150,130,150);
    Line(130,150,120,140);
    DrawPoly(SizeOf(CakmakShip) div SizeOf(PointType),CakmakShip);
    OutTextXY(115,90,'MAIN DECK');
    OutTextXY(10,200,'Length  Overall                = 119.02 m.');
    OutTextXY(10,220,'Length Between Perpendiculars  = 116.73 m.');
    OuttextXY(10,240,'Breadth, extreme               =  12.46 m.');
    OuttextXY(10,260,'Depth, Main Deck at Side @     =   6.97 m.');
    OuttextXY(10,280,'Depth, Main Deck, Lowest Point =   6.55 m.');
    OutTextXY(420,200,'Fuel Oil          =  723.23 tons');
    OutTextXY(420,220,'Lubricating Oil   =   11.12 tons');
    OutTextXY(420,240,'Potable Water     =   67.38 tons');
    OutTextXY(420,260,'Reserve Feed Water =   71.07 tons');
    OutTextXY(420,280,'Displacement      = 3543.60 tons');
    OutTextXY(55,160,'204');
    OutTextXY(105,160,'182');
    OutTextXY(205,160,'148');
    OutTextXY(335,160,'110 A&F');
    OutTextXY(475,160,'72');
    OutTextXY(575,160,'33');
    OutTextXY(654,160,'6');
    Readln;
end;
CloseGraph;
end;
end.
```

```
unit dm2;
    [* this unit is called by unit dm8 and used for draft functions *]

interface

 uses Crt,Graph,Dos,TestGr;
var
x1,y1,x2,y2,x3,y3,y4,i : integer;
Aft_draft,For_draft:real;
fac,slope : real;
Xasp,Yasp:word;
cokor:word;
same : boolean;
s3,s4:string;
 procedure dmscreen(s1,s2:string; var Aft_draft:real ; var For_draft: real);
 procedure main;
 implementation

procedure dmscreen(s1,s2:string;var Aft_draft:real; var For_draft:real);
[* s1 and s2 are strings are displayed in the screen window and output
 parameters Aft_draft and For_draft are read by the procedure *]
 var i: integer;
 begin
 clrscr;gotoxy(25,10);
 for i:= 1 to 31 do begin
 write(Chr(205));
 gotoxy(25+i,10);
 end;
 gotoxy(56,10); write(chr(187));
 gotoxy(56,11);
 for i:= 1 to 5 do begin
    write(Chr(186));
    gotoxy(56,10+i);
    end;
 gotoxy(56,14);write(chr(188));
```

```
gotoxy(55,14);
for i:= 1 to 31 do begin
 write(Chr(205));
 gotoxy(55-i,14);
 end;
 gotoxy(24,14);write(chr(200));
 gotoxy(24,13);
 for i:= 1 to 5 do begin
  write(chr(186));
  gotoxy(24,14-i);
  end;
  gotoxy(24,10);
  write(chr(201));
  gotoxy(24,12); write(chr(199));
  gotoxy(25,12);
     for i:= 1 to 24 do
      write(chr(196));
     write(chr(197));
     for i:= 1 to 6 do
     write(Chr(196));
     write(Chr(182));
     gotoxy(49,10);
     write(chr(209));
     gotoxy(49,11);
      for i:= 1 to 4 do begin
       write(chr(179));
       gotoxy(49,10+i);
       end;
       gotoxy(49,14); write(Chr(207));
       gotoxy(26,11); write(s1);
       gotoxy(26,13); write(s2);
       gotoxy(50,11); read(Aft_draft);
       gotoxy(50,13); read(For_draft);

end; [ procedure dmscreen]
```

```
procedure main;
begin
s3:=' AFTER  DRAFT MARK';
s4:='FORWARD DRAFT MARK';
dmscreen(s3,s4,Aft_draft,For_draft);
if not Test_Graph_Device(GraphDriver,ErrorCode,GraphMode) then
begin
 writeln('Program aborted');
 Halt(1);
 end
else
begin
GetAspectRatio(Xasp,Yasp);
y1 := (Round((17-Aft_draft)*40) + 60);
y2 := (Round((17-For_draft)*40) + 60);
same := abs(y1-y2) < 10;
SetTextJustify(CenterText,CenterText);
SetTextStyle(1,Horizdir,4);
for i:= 1 to 3 do
begin
OuttextXY(GetMaxX div 2,GetMaxY div 2,'TCG. MF. Cakmak');
Delay(1000);
ClearDevice;
end;

OutTextXY(GetmaxX div 2,15, 'DRAFT DIAGRAM AND FUNCTIONS');
SetTextStyle(2,0,4);

OutTextXY(50,50,'AFTER DRAFT MARKS');
OutTextXY(310,40,'DISPLACEMENT (Tons)');
OutTextXY(440,40,'IMMERSION (Tons/Inch)');
OutTextXY(650,50,'FORWARD DRAFT MARKS');
Line(60,60,60,340);
Line(320,47,320,340);
Line(450,60,450,340);
```

```
Line(660,60,660,340);
y3 := 60;
for i:= 1 to 30 do begin
 Line(55,y3,60,y3);
 Line(665,y3,660,y3);
 if y3 mod 40 = 0 then begin
   Line(50,y3,60,y3);
   Line(670,y3,660,y3);
   end;
 y3:=y3+10;
 end;
 y3:= 47;
 for i:= 1 to 22 do begin
   Line(315,y3,320,y3);
   if i mod 5 = 1 then
        Line(310,y3,320,y3);
        if i<6 then fac := 11.4;
        if ((i>=6) and (i<12)) then fac := 11.43;
        if ((i>=12) and (i<18)) then fac := 11.47
        else fac:= 11.50;
        y3:= Round(y3+fac);
   end;
   OutTextXY(290,105,'4000');
   OutTextXY(290,167,'3500');
   OutTextXY(290,220,'3000');
   OutTextXY(290,280,'2500');
   Line(450,79,455,79);
   OutTextXY(470,75,'29.0');
   Line(450,120,455,120);
   OutTextXY(470,115,'28.8');
   Line(450,146,455,146);
   OutTextXY(470,142,'28.6');
   Line(450,175,455,175);
   OutTextXY(470,171,'28.4');
   Line(450,200,455,200);
```

```
OutTextXY(470,197,'28.2');
Line(450,224,455,224);
OutTextXY(470,220,'28.0');
Line(450,245,455,245);
OutTextXY(470,242,'27.8');
Line(450,262,455,262);
OutTextXY(470,258,'27.6');
Line(450,276,455,276);
OutTextXY(470,272,'27.4');
Line(450,288,455,288);
OutTextXY(470,284,'27.2');
Line(450,296,455,296);
OutTextXY(470,292,'27.0');
OutTextXY(40,100,'XVI');        OutTextXY(680,100,'XVI');
OutTextXY(40,140,'XV');         OutTextXY(680,140,'XV');
OutTextXY(40,180,'XIV');        OutTextXY(680,180,'XIV');
OutTextXY(40,220,'XIII');       OutTextXY(680,220,'XIII');
OutTextXY(40,260,'XII');        OutTextXY(680,260,'XII');
OutTextXY(40,300,'XI');         OutTextXY(680,300,'XI');
OutTextXY(40,340,'X');          OutTextXY(680,340,'X');
delay(1000);
if not same then
Line(60,y1,660,y2)
else begin
cokor:= 15;
x1:= 60 ;
x2:= 660 ;
y4:= y1;
slope:= (y2-y1)/(x2-x1);
 while x1 < = x2 do
   begin
PutPixel(x1,y1,cokor);
x1:= x1 + 1;
y1:= Round((slope*x1)*(Xasp/Yasp)) + y4;
delay(10);
```

```
    end;
      end;
 Readln;
end;
Readln;
CloseGraph;
end;
end.
```

```
unit dm5;
[* this unit is called by  unit dm8 and used for liquid loading
  diagrams of the ship *]
interface
uses Graph,Dos,Crt,TestGr;
const
maxtank = 31;
var
i,j:integer;
found:boolean;
capacity,degrees,aft_draft,for_draft,aframe,fframe:string[ 7 ];
tankname:string[ 7 ];
explanation : string[ 25 ];
s : array[ 1..maxtank,1..7 ] of string[ 7 ];
MyCh:char;

procedure main;
implementation
procedure main;
begin
  s(.1,1.):='C-12F';   s(.1,2.):='38';   s(.1,3.):='2.0';
  s(.1,4.):='-2';      s(.1,5.):='+4';   s(.1,6.):='170';
  s(.2,1.):='C-11F';   s(.2,2.):='37';   s(.2,3.):='2.0';
  s(.2,4.):='-2';      s(.2,5.):='+4';   s(.2,6.):='170';
  s(.3,1.):='C-10F';   s(.3,2.):='54';   s(.3,3.):='2.9';
  s(.3,4.):='-2';      s(.3,5.):='+5';   s(.3,6.):='157';
  s(.4,1.):='C-9F';    s(.4,2.):='55';   s(.4,3.):='2.9';
  s(.4,4.):='-2';      s(.4,5.):='+5';   s(.4,6.):='157';
  s(.5,1.):='C-8F';    s(.5,2.):='24';   s(.5,3.):='1.4';
  s(.5,4.):='-1.0';    s(.5,5.):='+2';   s(.5,6.):='152';
  s(.6,1.):='C-7F';    s(.6,2.):='24';   s(.6,3.):='1.4';
  s(.6,4.):='-1';      s(.6,5.):='+2';   s(.6,6.):='152';
  s(.7,1.):='C-4F';    s(.7,2.):='11';   s(.7,3.):='0.8';
  s(.7,4.):='0';       s(.7,5.):='+1';   s(.7,6.):='148';
  s(.8,1.):='C-6F';    s(.8,2.):='61';   s(.8,3.):='0.0';
```

```
s(.8,4.):='-2';       s(.8,5.):='+5';      s(.8,6.):='152';
s(.9,1.):='C-1F';     s(.9,2.):='51';      s(.9,3.):='0.0';
s(.9,4.):='-1';       s(.9,5.):='+4';      s(.9,6.):='148';
s(.10,1.):='C-501J';  s(.10,2.):='21';     s(.10,3.):='0.0';
s(.10,4.):='-1';      s(.10,5.):='+2';     s(.10,6.):='157';
s(.11,1.):='B-9.5F';  s(.11,2.):='38';     s(.11,3.):='2.2';
s(.11,4.):='+2';      s(.11,5.):='+1';     s(.11,6.):='110F';
s(.12,1.):='B-9.25F'; s(.12,2.):='60';     s(.12,3.):='0.0';
s(.12,4.):='+3';      s(.12,5.):='+2';     s(.12,6.):='110F';
s(.13,1.):='B-10.5F'; s(.13,2.):='39';     s(.13,3.):='2.5';
s(.13,4.):='+2';      s(.13,5.):='+1';     s(.13,6.):='110F';
s(.14,1.):='B-9.75F';s(.14,2.):='19';      s(.14,3.):='0.0;
s(.14,4.):='+1';      s(.14,5.):='+1';     s(.14,6.):='110G';
s(.15,1.):='A-508F';  s(.15,2.):='16';     s(.15,3.):='0.7';
s(.15,4.):='+2';      s(.15,5.):='-1';     s(.15,6.):='60';
s(.16,1.):='A-507F';  s(.16,2.):='15';     s(.16,3.):='0.7';
s(.16,4.):='+2';      s(.16,5.):='-1';     s(.16,6.):='60';
s(.17,1.):='A-510F';  s(.17,2.):='14';     s(.17,3.):='0.7';
s(.17,4.):='+2';      s(.17,5.):='0';      s(.17,6.):='66';
s(.18,1.):='A-509F';  s(.18,2.):='14';     s(.18,3.):='0.7';
s(.18,4.):='+2';      s(.18,5.):='0';      s(.18,6.):='66';
s(.19,1.):='A-506F';  s(.19,2.):='19';     s(.19,3.):='0.3';
s(.19,4.):='+3';      s(.19,5.):='-1';     s(.19,6.):='48';
s(.20,1.):='A-505F';  s(.20,2.):='12';     s(.20,3.):='0.2';
s(.20,4.):='+2';      s(.20,5.):='-1';     s(.20,6.):='48';
s(.21,1.):='B-14W';   s(.21,2.):='19';     s(.21,3.):='1.7';
s(.21,4.):='0';       s(.21,5.):='+1';     s(.21,6.):='120.5';
s(.22,1.):='B-12W';   s(.22,2.):='21';     s(.22,3.):='1.8';
 s(.22,4.):='+1';      s(.22,5.):='+1';     s(.22,6.):='110A';
s(.23,1.):='B-11W';   s(.23,2.):='21';     s(.23,3.):='1.8';
s(.23,4.):='+1';      s(.23,5.):='+1';     s(.23,6.):='110A';
s(.24,1.):='B-13W';   s(.24,2.):='19';     s(.24,3.):='1.7';
s(.24,4.):='0';       s(.24,5.):='+1';     s(.24,6.):='120.5';
s(.25,1.):='B-10W';   s(.25,2.):='16';     s(.25,3.):='1.2';
s(.25,4.):='+1';      s(.25,5.):='0';      s(.25,6.):='82.5';
```

```
  s(.26,1.):='B-8W';    s(.26,2.):='13';    s(.26,3.):='0.9';
  s(.26,4.):='+1';      s(.26,5.):='0';     s(.26,6.):='72';
  s(.27,1.):='B-7W';    s(.27,2.):='13';    s(.27,3.):='0.9';
  s(.27,4.):='+1';      s(.27,5.):='0';     s(.27,6.):='72';
  s(.28,1.):='B-9W';    s(.28,2.):='16';    s(.28,3.):='1.2';
  s(.28,4.):='+1';      s(.28,5.):='0';     s(.28,6.):='82,5';
  s(.29,1.):='C-5F';    s(.29,2.):='5';     s(.29,3.):='0.4';
  s(.29,4.):='0';       s(.29,5.):='0';     s(.29,6.):='148';
  s(.30,1.):='A-3F';    s(.30,2.):='55';    s(.30,3.):='0';
  s(.30,4.):='+7';      s(.30,5.):='-2';    s(.30,6.):='60';
  s(.31,1.):='A-4F';    s(.31,2.):='47';    s(.31,3.):='0';
  s(.31,4.):='+6';      s(.31,5.):='-1';    s(.31,6.):='66';
  s(.1,7.):='182';      s(.2,7.):='182';    s(.3,7.):='170';
  s(.4,7.):='170';      s(.5,7.):='157';    s(.6,7.):='157';
  s(.7,7.):='152';      s(.8,7.):='157';    s(.9,7.):='152';
  s(.10,7.):='170';     s(.11,7.):='110G';  s(.12,7.):='110G';
  s(.13,7.):='110G';    s(.14,7.):='110A';  s(.15,7.):='66';
  s(.16,7.):='66';      s(.17,7.):='72';    s(.18,7.):='72';
  s(.19,7.):='60';      s(.20,7.):='60';    s(.21,7.):='130.5';
  s(.22,7.):='120.5';   s(.23,7.):='120.5'; s(.24,7.):='130.5';
  s(.25,7.):='92.5';    s(.26,7.):=82.5';   s(.27,7.):='82.5';
  s(.28,7.):='92.5';    s(.29,7.):='152';   s(.30,7.):='66';
                        s(.31,7.):='72.5';
repeat
clrscr;
gotoxy(20,12); write(Chr(201));
for j:= 1 to 39 do
 write(chr(205));
write(chr(187));
gotoxy(60,13); write(chr(186));
gotoxy(60,14); write(chr(188));
gotoxy(21,14);
for j:=1 to 39 do
 write(chr(205));
 gotoxy(20,14); write(chr(200));
```

```
 gotoxy(20,13); write(chr(186));
 gotoxy(21,13); write(' ENTER THE NAME OF THE TANK ');
 gotoxy(52,13); write(chr(179));
 gotoxy(52,12); write(chr(209));
 gotoxy(52,14); write(chr(207));
 gotoxy(53,13); readln(tankname);
 found:= false;
 i:= 1;
repeat
  if s[ i,1 ]= tankname then found:= true
  else begin
  i:= i+ 1;
  found:= false;
  end;
  until (i> maxtank) or (found);
  capacity:= s[ i,2 ];
  degrees:= s[ i,3 ];
 for_draft:= s[ i,4 ];
 aft_draft:= s[ i,5 ];
 fframe:= s[ i,6 ];
 aframe:= s[ i,7 ];
 if ((s[ i,1 ]= 'B-14W') or (s[ i,1 ]= 'B-13W')) or
                    ((s[i,1]= 'B-10W') or (s[i,1]= 'B-9W'))  then
  explanation:= 'RESERVE FEED TANK'
 else if
      ((s[i,1]= 'B-12W') or (s[i,1]= 'B-11W')) or
                         ((s[i,1]= 'B-7W') or (s[i,1]= 'B-8W')) then
  explanation:= 'FRESH WATER TANK'
else if (s[ i,1 ]= 'C-501J') then explanation:= 'DIESEL OIL TANK'
else if ((s[ i,1 ]= 'C-1F') or (s[ i,1 ]= 'C-6F')) then
    explanation:= 'AFT SERVICE TANK'
else if ((s[ i,1 ]= 'A-3F') or (s[ i,1 ]= 'A-4F')) then
    explanation:= 'FWRD  SERVICE TANK'
else if ((s[ i,1 ]= 'A-501W') or (s[ i,1 ]= 'A-1W')) then
     explanation :=  'FORWARD PEAK TANK'
```

```
else if ((s[ i,1 ]= 'C-312W') or (s[ i,1 ]= 'C-311A')) then
   explanation:= 'AFTER PEAK TANK'
else explanation:= 'FUEL OIL BALLAST TANK';
if not Test_Graph_Device(GraphDriver,ErrorCode,GraphMode) then
begin
writeln('Program aborted');
Halt(1);
end
else
   begin
rectangle(180,80,540,220);
OutTextXY(190,85,capacity+' tons');
OutTextXY(510,85,degrees);
OutTextXY(460,210,for_draft+' inches');
OutTextXY(190,210,aft_draft+' inches');
OutTextXY(340,150,s[ i,1 ]);
rectangle(155,225,575,250);
OutTextXY(170,230,aframe);
OutTextXY(530,230,fframe);
OutTextXY(290,170,explanation);
Rectangle(20,20,700,328);
FLOODFILL(25,25,15);
Readln;
CloseGraph;
      end;
 end
else  begin
 clrscr;
 for j:= 1 to 3 do begin
DELAY(500);
gotoxy(35,12);
write('NO SUCH TANK');
DELAY(500);
clrscr;
 end;
```

```
 end;
gotoxy(32,12); write(' DO YOU WANT TO QUIT..? ');
gotoxy(43,14); readln(MyCh);
found:= false;
until Upcase(MyCh)= 'Y';
gotoxy(20,10);
end;
end.
```

```
unit dm6;
[* This unit is called by unit dm8 and used for fire_fighting implementation *]

interface

uses Dos,Crt,Graph,dm2;
type
 fireplug = record
    deck : integer;
       frame : integer ;
       side:integer;
       location : string[30];
       size_type : string[12];
       end;
 var
 found1,found2,found3:boolean;
 xi,xf,yi,yf:integer;
 b:longint;
 firetype:char;
 fire_stations : array[ 1..37 ] of fireplug;
 virtual_station : array [ 1..37 ] of fireplug;
 i,j,l,k,m:integer;
 dno,fno : real;
 d2no,f2no : integer;
 f: text;
 tg:char;

procedure fire_type;
procedure dm6_screen(k,l,deck,frame,side:integer; location:string);
procedure mybox(xi,yi,xf,yf:integer);
procedure main;

implementation
procedure fire_type;
[* draws a box for fire_type input *]
```

```
var j:integer;
      hc:CHAR;
 begin
 delay(2000);
 gotoxy(24,12); write(chr(201));
 for j:= 1 to 30 do write(chr(205));
 write(chr(187));
 gotoxy(55,13); write(chr(186));
 gotoxy(55,14); write(chr(188));
 gotoxy(24,14); write(chr(200));
 for j:= 1 to 30 do write(chr(205));
 gotoxy(24,13); write(chr(186));
 gotoxy(27,13); write('ENTER THE FIRETYPE');
 gotoxy(51,12); write(chr(209));
 gotoxy(51,13); write(chr(179));
 gotoxy(51,14); write(chr(207));
 gotoxy(27,13); write('ENTER THE FIRE TYPE');
 end;
procedure dm6_screen(k,l,deck,frame,side:integer;location:string);
[* displays the closest available fire plugs locations *]
begin
gotoxy(k,l);
writeln(deck,'-',frame,'-',side,'  ',location);
l:=l+1;
end;
procedure mybox(xi,yi,xf,yf:integer);
[* draws a double window on the screen coordinates specified as
   xinitial, yinitial,xfinal,yfinal *]
var j:integer;
begin
  gotoxy(xi,yi); write(chr(201));
  for j:= 1 to (xf-xi-1) do write(chr(205));
  write(chr(187));
  gotoxy(xf,yi+1); write(chr(186));
  gotoxy(xf,yf); write(chr(188));
```

```
  gotoxy(xi,yf); write(chr(200));
  for j:= 1 to (xf-xi-1) do write(chr(205));
  gotoxy(xi,yi+1); write(chr(186));
  end;
procedure main;
begin
b:= 0;
for j:= 1 to 37 do
virtual_station[j].deck:= 0;
clrscr;
i:= 1;
Assign(f,'fire.txt');
reset (f);
while not eof(f) do begin
 with fire_stations[i] do begin
 readln(f,deck,frame,side,location);
 end;
 i:= i+1;
 end;
 Close(f);
 clrscr;
 fire_type;
 gotoxy(52,13);  read(firetype);
 if ((firetype= Chr(65)) or (firetype= Chr(97))) then begin
 clrscr;
 dm2.dmscreen('ENTER THE DECK NUMBER','ENTER THE FRAME
NUMBER',dno,fno);
 clrscr;
 gotoxy(20,3); write(chr(201));
 for j:= 1 to 36 do
  write(chr(205)); write(chr(187));
  gotoxy(57,4);
  for j:= 1 to 21 do begin
   write(chr(186));
   gotoxy(57,4+j);
```

```
end;
gotoxy(57,25); write(chr(188));
gotoxy(20,25);write(chr(200));
for j:= 1 to 36 do
write(chr(205));
gotoxy(20,4);
for j:= 1 to 21 do begin
   write(chr(186));
   gotoxy(20,4+j);
   end;
   gotoxy(24,4); write(' VALVE ');
   gotoxy(40,4); write('LOCATION');
   gotoxy(20,5);
    for j:= 1 to 20 do begin
    write(chr(199));
     for k:= 1 to 12 do write(chr(196));
     write(chr(197));
     for k:= 1 to 23 do write(chr(196));
     write(chr(182));
     j:=j+1;
     gotoxy(20,j+5);
     end;
  gotoxy(33,3); write(chr(209));
    gotoxy(33,4);
  for j:= 1 to 20 do begin
     write(chr(179));
     j:=j+1;
     gotoxy(33,j+4);
     end;
     mybox(60,3,69,5);
     gotoxy(60,3); write(chr(201));
     for j:= 1 to 8 do write(chr(205)); write(chr(187));
     gotoxy(69,4); write(chr(186));
     gotoxy(69,5); write(chr(188));
     gotoxy(60,5); write(chr(200));
```

```
        for j:=  1 to 8 do write(chr(205));
        gotoxy(60,4); write(chr(186));

d2no:= Trunc(dno);
f2no:= Trunc(fno);
found1:= false;
found2:= false;
found3:= false;
if dno = 5 then begin
     if ((fno < = 92) and (fno > = 72)) then begin
        l:= 6;
        for j:= 1 to 3 do begin
        with fire_stations[j] do begin
         found1:= true;
         found2:= true;
         found3:= true;
         dm6_screen(24,l,deck,frame,side,location);
         l:= l+ 2;
         end;
         end;
         for j:= 28 to 29 do begin
         with fire_stations[j] do begin
         dm6_screen(24,l,deck,frame,side,location);
         l:= l+ 2;
         end;
         end;
         end;
     if ((fno < = 110) and (fno > = 93)) then begin
     l:= 6;
     found1:= true;
     found2:= found1;
     found3:= found2;
        for j:= 4 to 6 do begin
         with fire_stations[j] do begin
             dm6_screen(24,l,deck,frame,side,location);
```

```
        l:=l+2;
        end;
        end;
    for j:=30 to 31 do begin
        with fire_stations[j] do begin
        dm6_screen(24,l,deck,frame,side,location);
        l:=l+2;
        end;
        end;
        end;
 if ((fno<130) and (fno>110)) then begin
    l:=6;
    found1:=true;
    found2:=true;
    found3:=true;
    for j:=7 to 9 do begin
     with fire_stations[j] do begin
     dm6_screen(24,l,deck,frame,side,location);
     l:=l+2;
     end;
     end;
     for j:=30 to 34 do begin
     with fire_stations[j] do begin
     dm6_screen(24,l,deck,frame,side,location);
     l:=l+2;
     end;
     end;
     end;

 if ((fno<=148) and (fno>=130)) then begin
l:=6;
    found1:=true;
    found2:=true;
    found3:=true;
     for j:=10 to 13 do
```

```
        begin
         with fire_stations[j] do begin
         dm6_screen(24,l,deck,frame,side,location);
         l:=l+2;
         end;
         end;
         for j:= 30 to 34 do begin
         with fire_stations[j] do begin
         dm6_screen(24,l,deck,frame,side,location);
         l:=l+2;
         end;
         end;
         end;
    end
      else begin
      for j:= 1 to 37 do begin
       if abs(fire_stations[j].frame - fno) <= 13 then
     virtual_station[j]:=fire_stations[j];
     end;
       end;
       l:=6;
       for j:= 1 to 37 do begin
       if virtual_station[j].deck<>0 then begin
        with virtual_station[j] do begin
        dm6_screen(24,l,deck,frame,side,location);
        l:=l+2;
        end;
        end;
        end;
gotoxy(33,4);
for j:= 1 to 20 do begin
      write(chr(179));
      j:=j+1;
      gotoxy(33,j+4);
      end;
```

```
   gotoxy(33,24); write(chr(179));
gotoxy(33,25); write(chr(207));
   repeat
   if (fno < = 72) then begin
   gotoxy(28,1); write(' FORWARD REPAIR TEAM ');
   delay(490);
gotoxy(28,1); write('  STOP VENTILATION    ');
   delay(500);
   end;
   if ((fno > 72) and (fno < = 148)) then begin
    gotoxy(28,1); write(' MIDSHIP REPAIR TEAM');
    DELAY(490);
    gotoxy(28,1); write('  STOP VENTILATION    ');
    delay(500);
    end;
    if fno > 148 then begin
     gotoxy(28,1); write(' AFTER  REPAIR  TEAM');
     DELAY(490);
     gotoxy(28,1); write('  STOP VENTILATION    ');
     delay(500);
     end;
   b: = b + 1;
   gotoxy(64,4); write(b);
   mybox(60,7,75,9); gotoxy(61,8);
   write(' COMLINE 2JZ');

until KeyPressed;
   clrscr;
   tg: = #3;
mybox(32,11,50,13);
   gotoxy(35,12);
   write('SCORE = = = > ',b);
delay(3000);
   end
   else if ((firetype = chr(66)) or (firetype = chr(98))) then begin
```

```
      b:=0;
      repeat
      clrscr;
      delay(500);
      mybox(23,7,59,9);
      gotoxy(24,8);
       writeln('          DO NOT USE WATER          ');
       DELAY(1000);
       mybox(23,10,59,12);
       gotoxy(24,11);
       writeln(' CO2 OR DRY CHEMICAL AGENTS');
       delay(1000);
       mybox(23,13,60,15);
       gotoxy(24,14);
       WRITELN(' CUT OFF LIGHTING CIRCUIT BREAKERS');
       delay(1000);
       mybox(23,16,60,18); gotoxy(24,17);
       write('  COMMUNICATION LINE IS = = > 2JZ ');
       DELAY(1000);
       b:=b+5;
       mybox(35,19,45,21);
       gotoxy(39,20);
       write(b);
       delay(500);
       until KeyPressed;
       end
       else if firetype=#116 then begin
      writeln(' NO SUCH FIRETYPE');
      delay(500);
      clrscr;
    end;
end;
 end.
```

```
unit dm8;
[* this unit is the main part of the damage control part.This unit calls
   all other units.It is also called by outmenu.pas turbo pascal unit
   and executed when user selects option 3 in the main menu. *]
interface
uses Crt,dm1,dm2,dmb3,dmf3,dmw3,dm6,dm5;
var
i,j,k:integer;
c1,c2,c3,c4,c5,c6,c7,c8:boolean;
bc,tc:integer;
Ch:char;
currenti:integer;
procedure bir(i:integer; var j:integer :
          bc,tc: integer ; var current1 : boolean);
procedure iki(i:integer; var j:integer :
          bc,tc: integer : var current2 : boolean);
procedure uc( i: integer; var j:integer;
           bc,tc : integer; var current3 : boolean);
procedure dort(i:integer; var j:integer;
           bc,tc : integer; var current4: boolean);
procedure bes( i: integer; var j:integer;
           bc,tc : integer; var current5:boolean);
procedure  alti(i:integer; var j:integer;
           bc,tc : integer; var current6: boolean);
procedure yedi( i: integer; var j : integer;
           bc,tc : integer; var current7:boolean);
procedure sekiz(i:integer; var j: integer;
           bc,tc : integer; var current8: boolean);
procedure mscreen(current1,current2,current3,current4,current5,current6,
             current7,current8 : boolean;
                    var j: integer; var k:integer);
procedure box2( xb,yb,xe,ye:integer);
procedure main;
implementation
procedure bir( i: integer; var j: integer; bc,tc : integer;
```

```
            var current1 : boolean);
[* procedures bir through sekiz are called by procedure mscreen
  and used to determine which option should be highlighted in
  the wrap-around & pull-down menu implementation *]
begin
i:= 30; j:= 10;
if current1 then begin
bc:= 7; tc:= 0;
end
else begin
bc:= 0; tc:= 7;
end;
TextBackGround(bc); TextColor(tc);
gotoxy(i,j); write('1. SHIP GENERAL INFO ');
end;
procedure iki(i:integer; var j:integer ;
bc,tc:integer; var current2 : boolean);
begin
i:= 30; j:= 11;
if current2 then begin
bc:= 7; tc:= 0 ;
end
else begin
   bc:= 0; tc:= 7;
     end;
        TextBackGround(bc); Textcolor(tc);
        gotoxy(i,j); write('2. DRAFT DIAGRAM ');
        end;
procedure uc(i:integer; var j:integer ;
bc,tc: integer; var current3:boolean);
begin
i:= 30; j:= 12;
if current3 then begin
          bc:= 7; tc:= 0;
               end
```

```
else begin
        bc:=0; tc:=7;
   end;
      TextBackGround(bc); TextColor(tc);
      gotoxy(i,j); write('3. SHOW  BALLAST  TANKS ');
      end;
procedure dort(i:integer; var j:integer;
bc,tc:integer ; var current4: boolean);
begin
i:=30; j:=  13;
if current4 then begin
        bc:=7; tc:=0 ;
            end
else begin
        bc:=0; tc:=7 ;
      end;
      TextBackGround(bc); TextColor(tc);
      gotoxy(i,j); write('4. SHOW  FUEL  TANKS');
      end;
procedure bes(i:integer; var j:integer;
bc,tc:integer; var current5:boolean);
begin
i:=30; j:=14;
if current5 then begin
        bc:=7; tc:=0;
            end
else begin
            bc:=0; tc:=7;
       end;
       TextBackGround(bc); Textcolor(tc);
       gotoxy(i,j); write('5. SHOW  WATER  TANKS');
       end; procedure alti(i:integer; var j:integer; bc,tc:integer; var current6:boolean); var
cunt:integer;
begin
i:=30; j:=15;
```

```
if current6 then begin
                bc:=7; tc:=0; cunt:=cunt+1;
                end
else begin
               bc:=0; tc:=7;
    end;
       TextBackGround(bc); Textcolor(tc);
       gotoxy(i,j); write('6. FIRE  FIGHTING EXERCISE ');
       if cunt mod 3 < > 0 then write('CONFIRM ');
       end;
procedure yedi (i: integer; var j : integer;
            bc,tc:integer; var current7:boolean);
begin
i:=30; j:=  16;
if current7 then begin
  bc:=7; tc:=0;
  end
  else begin
  bc:=0; tc:=7;
  end;
  TextBackGround(bc); Textcolor(tc);
  gotoxy(i,j); write('7. LIQUID  LOADING  DIAGRAM');
  end;
procedure sekiz(i:integer; var j:integer;
bc,tc:integer; var current8:boolean);
begin
i:=30; j:=17;
if current8 then begin
                bc:=7; tc:=0;
                end
else begin
          bc:=0; tc:=7;
      end;
      TextBackGround(bc); Textcolor(tc);
      gotoxy(i,j); write('8. EXIT  THIS PROGRAM ');
```

```
    end;
procedure mscreen(current1,current2,current3,current4,current5,
            current6,current7,current8 : boolean;
                    var j: integer; var k: integer);
[* this procedure calls previous ones bir through sekiz by turn
  and handles the pull-down & wrap-around type menu *]
begin
  bir(i,j,bc,tc,current1);
  iki(i,j,bc,tc,current2);
  uc(i,j,bc,tc,current3);
  dort(i,j,bc,tc,current4);
  bes(i,j,bc,tc,current5);
  alti(i,j,bc,tc,current6);
  yedi(i,j,bc,tc,current7);
  sekiz(i,j,bc,tc,current8);
  if current1 then k:=10;
  if current2 then k:=11;
  if current3 then k:=12;
  if current4 then k:=13;
  if current5 then k:=14;
  if current6 then k:=15;
  if current7 then k:=16;
  if current8 then k:=17;
end;
procedure box2(xb,yb,xe,ye:integer);
[* draws a single window on the screen coordinates specified as
   xbeginning,ybeginning,xend,yend. *]
var j:integer;
begin
gotoxy(xb,yb); write(chr(218));
for j:=1 to xe-1 do write(chr(196)); write(chr(191));
gotoxy(xe+1,yb+1);
for j:=1 to ye do begin
write(chr(179));
gotoxy(xe+1,yb+j);
```

```
end;
gotoxy(xb,yb+1);
for j:= 1 to ye-1 do begin
write(chr(179));
gotoxy(xb,yb+j);
end;
gotoxy(xb,ye); write(chr(192));
for j:= 1 to xe-1 do write(chr(196));
write(chr(217));
end;
procedure main;
begin
repeat
dm6.mybox(20,20,60,22);
gotoxy(30,21); write(' USE '+CHR(24)+ ' AND ' +CHR(25) +' ARROWS');
clrscr;
c1:= true; c2:= false; c3:= false;c4:= false;c5:= false;c6:= false;c7:= false;
c8:= false;
j:= 10;
 repeat
 box2(1,1,78,24);
 gotoxy(28,5); write(' DAMAGE CONTROL CENTER');
 gotoxy(26,6);
 for j:= 1 to 29 do write(chr(196));
 dm6.mybox(20,20,60,22);
 gotoxy(30,21); write(' USE '+chr(24)+ ' AND ' +chr(25) +' ARROWS');
 mscreen(c1,c2,c3,c4,c5,c6,c7,c8,j,k);
   Ch:= ReadKey;
  if Ch=#80 then
   k:=k+1;
   if Ch=#72 then
   k:=k-1;
   if k=18 then begin k:=10; j:=10; end;
   if k=9 then begin k:=17; j:=17; end;
   if (( k=10 ) OR (ch=#49)) then begin
```

```
c1:=true;    c2:=false;  c3:=false;  c4:=false;
c5:=false;   c6:=false;  c7:=false;  c8:=false;
mscreen(c1,c2,c3,c4,c5,c6,c7,c8,j,k);
end;
if ((k=11) or (Ch=#50)) then begin
c2:=true;   c1:=false;   c3:=false;   c4:=false;
c5:=false;  c6:=false;   c7:=false;   c8:=false;
mscreen(c1,c2,c3,c4,c5,c6,c7,c8,j,k);
end;
if ((k=12) or (Ch=#51)) then begin
c3:=true;  c1:=false;  c2:=false;  c4:=false;
c5:=false; c6:=false;  c7:=false;  c8:=false;
mscreen(c1,c2,c3,c4,c5,c6,c7,c8,j,k);
end;
if ((k=13) or (Ch=#52)) then begin
c4:=true;   c1:=false;   c2:=false;   c5:=false;
c3:=false;  c6:=false;   c7:=false;   c8:=false;
mscreen(c1,c2,c3,c4,c5,c6,c7,c8,j,k);
end;
if ((k=14) or (Ch=#53)) then begin
c5:=true;  c1:=false;   c2:=false;  c3:=false;
c4:=false; c6:=false;   c7:=false;  c8:=false;
mscreen(c1,c2,c3,c4,c5,c6,c7,c8,j,k);
end;
if ((k=15) or (Ch=#54)) then begin
c6:=true;  c1:=false;  c3:=false;  c4:=false;
c5:=false; c2:=false;  c7:=false;  c8:=false;
mscreen(c1,c2,c3,c4,c5,c6,c7,c8,j,k);
end;
if ((k=16) or (Ch=#55)) then begin
c7:=true;  c1:=false;  c2:=false;  c3:=false; c4:=false;
c5:=false; c6:=false;  c8:=false;
mscreen(c1,c2,c3,c4,c5,c6,c7,c8,j,k);
end;
if ((k=17) or (Ch=#56)) then begin
```

```
    c8:= true;  c1:=  false; c2:= false; c3:= false;
    c4:= false; c5:= false;  c6:= false; c7:= false;
    TextBackGround(0);
    TextColor(7);
    end;
    until Ch=#13;
    clrscr;
    gotoxy(i,j);
     if k=10 then dm1.main;
        if k=11 then dm2.main;
        if k=12 then dmb3.main;
        if k=13 then dmf3.main;
        if k=14 then dmw3.main;
        if k=15 then dm6.main;
        if k=16 then dm5.main;
until k=17;
gotoxy(20,20);
end;
end.
```

```
unit dmb3;
[* this unit is called by unit dm8 and used to draw the fuel oil
  ballast tanks of the ship in the graphics mode *]
interface
uses Graph,Dos,TestGr;
const
C501 : array[ 1..5 ] of PointType = ((x:45; y:134),
                                     (x:96; y:134),
                                     (x:96; y:166),
                                     (x:45;y:166),
                                     (x:45; y:134));
 C12_10_8_4_2 : array[ 1..5 ] of PointType = ((x:10; y:125),
                                          (x:130; y:104),
                                          (x:130; y:134),
                                          (x:10;  y:134),
                                          (x:10;  y:125));
C11_9_7_5_3 : array[ 1..5 ] of PointType = ((x:10; y:166),
                                        (x:130; y:166),
                                        (x:130; y:196),
                                        (x:10;  y:175),
                                        (x:10;  y:166));
BBS : array[ 1..5 ] of PointType =          ((x:275; y:100),
                                       (x:306; y:101),
                                       (x:306; y:199),
                                       (x:275; y:200),
                                       (x:275; y:100));

A510_508 : array[ 1..5 ] of PointType = ((x:456; y:111),
                                    (x:509; y:116),
                                    (x:509; y:137),
                                    (x:456; y:137),
                                   (x:456; y:111));
A509_507 : array[ 1..5 ] of PointType = ((x:456; y:166),
                                    (x:509; y:166),
                                    (x:509; y:184),
```

```
                                (x:456; y:189),
                                (x:456; y:166));

A506_505 : array[ 1..5 ] of PointType = ((x:509; y:125),
                                (x:559; y:129),
                                (x:559; y:171),
                                (x:509; y:175),
                                (x:509; y:125));
LoadingShip : array [ 1..46 ] of PointType = (( x:10 ; y:125),
                                ( x:45 ; y:113),
                                ( x:96 ; y:108),
                                ( x:114; y:105),
                                ( x:130; y:104),
                                ( x:130; y:102),
                                ( x:198; y:101),
                                ( x:236; y:100);
                                ( x:275; y:100);
                                ( x:278; y:100);
                                ( x:306; y:101);
                                ( x:374; y:103);
                                ( x:412; y:106);
                                ( x:453; y:109);
                                ( x:453; y:111);
                                ( x:456; y:113),
                                ( x:475; y:113),
                                ( x:509; y:116),
                                ( x:509; y:125),
                                ( x:554, y:129),
                                ( x:682, y:145),
                                ( x:690; y:147),
                                ( x:695; y:150),
                                ( x:690; y:153),
                                ( x:682; y:155),
                                ( x:554; y:171),
                                ( x:509; y:175),
```

```
                    ( x:509; y:184),
                    ( x:475; y:187),
                    ( x:456; y:187),
                    ( x:453; y:189),
                    ( x:453; y:191),
                    ( x:412; y:194),
                    ( x:374; y:197),
                    ( x:306; y:199),
                    ( x:278; y:200),
                    ( x:275; y:200),
                    ( x:236; y:200),
                    ( x:198; y:199),
                    ( x:130; y:198),
                    ( x:130; y:196),
                    ( x:114; y:195),
                    ( x:96;  y:192),
                    ( x:45;  y:187),
                    ( x:10;  y:175),
                    ( x:10;  y:125));

procedure main;
implementation
procedure main;
begin
if not Test_Graph_Device(GraphDriver,ErrorCode,GraphMode) then
 begin
  writeln('Program aborted');
  Halt(1);
  end
else
begin

 DrawPoly(SizeOf(LoadingShip) div SizeOf(PointType),LoadingShip);
 Line(10,134,130,134);
 Line(10,166,130,166);
```

```
Line(45,113,45,187);
Line(96,108,96,192);
line(114,105,114,195);
Line(130,102,130,198);
Line(198,101,198,199);
Line(236,100,236,115);
Line(236,185,236,200);
Line(275,100,275,200);
Line(278,100,278,200);
Line(306,101,306,199);
Line(278,131,306,131);
Line(278,168,306,168);
Line(374,103,374,197);
Line(412,106,412,119);
Line(412,181,412,194);
Line(453,109,453,191);
Line(456,111,456,189);
Line(479,113,479,187);
Line(509,116,509,184);
Line(559,129,559,171);
line(678,145,678,155);
Line(690,147,690,153);
Line(695,148,695,152);
Line(198,115,275,115);
Line(198,185,275,185);
Line(374,116,453,122);
Line(374,184,453,178);
Line(456,137,509,137);
Line(456,166,509,166);
Line(509,159,524,159);
Line(524,159,524,150);
Line(524,150,559,150);
OutTextXY(390,150,'B-1-1');
OutTextXY(220,150,'B-3-1');
OutTextXY(320,150,'B-2');
```

```
OutTextXY(150,150,'B-4');
OutTextXY(550,175,'48');
OutTextXY(504,188,'60');
OutTextXY(448,196,'72');
OutTextXY(370,204,Chr(57)+Chr(50)+Chr(171));
OutTextXY(270,205,'110');
OutTextXY(193,204,'130');
OutTextXY(125,203,'148');
OutTextXY(91,197,'157');
OutTextXY(40,192,'170');
OutTextXY(5,180,'182');
SetfillStyle(4,0);
FillPoly(SizeOf(C12_10_8_4_2) div SizeOf(PointType),C12_10_8_4_2);
FillPoly(SizeOf(C11_9_7_5_3) div SizeOf(PointType),C11_9_7_5_3);
FillPoly(SizeOf(BBS) div SizeOf(PointType),BBS);
FillPoly(SizeOf(A510_508) div SizeOf(PointType),A510_508);
FillPoly(SizeOf(A509_507) div SizeOf(PointType),A509_507);
FillPoly(SizeOf(A506_505) div SizeOf(PointType),A506_505);
SetFillStyle(7,0);
Fillpoly(SizeOf(C501) div SizeOf(PointType),C501);
Line(10,134,130,134); [ aft ballast tanks separator ]
Line(10,166,130,166);
Line(45,113,45,187);
Line(96,108,96,192);
Line(114,105,114,195);
Line(130,102,130,198);
Line(278,100,278,200); [ Mid ballast tanks separator ]
Line(306,101,306,199);
Line(278,131,306,131);
Line(278,168,306,168);
Line(456,111,456,191); [ Forw. ballast tanks separator ]
Line(475,113,475,187);
Line(509,116,509,184);
Line(456,111,509,116);
Line(456,166,509,166);
```

```
  Line(509,159,524,159);
  Line(524,159,524,150);
  Line(524,150,554,150);

  SetTextStyle(1,0,3);
  SetTextJustify(CenterText,CenterText);
  OutTextXY(320 , 250,'TCG M.F.CAKMAK  D351');
  OutTextXY(300,280,'CONFIGURATIONAL  PLAN  ,  FUEL  OIL  BALLAST
TANKS');
 Readln;
 end;
 CloseGraph;
  end;
 end.
```

```
unit dmf3;
[* this unit is called by unit dm8 and used to draw the fuel oil
  service tanks of the ship in the graphics mode *]
interface
uses Graph,Dos,TestGr;
const
 A34F : array[ 1..5 of PointType = ((x:456; y:137),
                                    (x:509; y:137),
                                    (x:509; y:166),
                                    (x:456; y:166),
                                    (x:456; y:137));
C16F : array[ 1..5 ] of PointType = ((x:96; y:134),
                                    (x:130; y:134),
                                    (x:130; y:166),
                                    (x:96; y:166),
                                    (x:96; y:134));

LoadingShip : array [ 1..46 ] of PointType = (( x:10 ; y:125),
                                    ( x:45 ; y:113),
                                    ( x:96 ; y:108),
                                                ( x:114; y:105),
                                                ( x:130; y:104),
                                                ( x:130; y:102),
                                                ( x:198; y:101),
                                                ( x:236; y:100),
                                                ( x:275; y:100),
                                                ( x:278; y:100),
                                                ( x:306; y:101),
                                                ( x:374; y:103),
                                                ( x:412; y:106),
                                                ( x:453; y:109),
                                                ( x:453; y:111),
                                                ( x:456; y:113),
                                                ( x:475; y:113),
                                                ( x:509; y:116),
```

```
                                        ( x:509; y:125),
                                        ( x:554; y:129),
                                        ( x:682; y:145),
                                        ( x:690; y:147),
                                        ( x:695; y:150),
                                        ( x:690; y:153),
                                        ( x:682; y:155),
                                        ( x:554; y:171),
                                        ( x:509; y:175),
                                        ( x:509; y:184),
                                        ( x:475; y:187),
                                        ( x:456; y:187),
                                        ( x:453; y:189),
                    ( x:453; y:191),
                                        ( x:412; y:194),
                                        ( x:374; y:197),
                                        ( x:306; y:199),
                                        ( x:278; y:200),
                                        ( x:275; y:200),
                                        ( x:236; y:200),
                                        ( x:198; y:199),
                                        ( x:130; y:198),
                                        ( x:130; y:196),
                                        ( x:114; y:195),
                                        ( x:96;  y:192),
                    ( x:45;  y:187),
                                        ( x:10;  y:175),
                                        ( x:10;  y:125));

procedure main;
implementation
procedure main;
begin
if not Test_Graph_Device(GraphDriver,ErrorCode,GraphMode) then
 begin
```

```
 writeln('Program aborted');
 Halt(1);
 end
else
begin
DrawPoly(SizeOf(LoadingShip) div SizeOf(PointType),LoadingShip);
Line(10,134,130,134);
Line(10,166,130,166);
Line(45,113,45,187);
Line(96,108,96,192);
line(114,105,114,195);
Line(130,102,130,198);
Line(198,101,198,199);
Line(236,100,236,115);
Line(236,185,236,200);
Line(275,100,275,200);
Line(278,100,278,200);
Line(306,101,306,199);
Line(278,131,306,131);
Line(278,168,306,168);
Line(374,103,374,197);
Line(412,106,412,119);
Line(412,181,412,194);
Line(453,109,453,191);
Line(456,111,456,189);
Line(479,113,479,187);
Line(509,116,509,184);
Line(559,129,559,171);
line(678,145,678,155);
Line(690,147,690,153);
Line(695,148,695,152);
Line(198,115,275,115);
Line(198,185,275,185);
Line(374,116,453,122);
Line(374,184,453,178);
```

```
 Line(456,137,509,137);
 Line(456,166,509,166);
 Line(509,159,524,159);
 Line(524,159,524,150);
 Line(524,150,559,150);
 OutTextXY(390,150,'B-1-1');
 OutTextXY(220,150,'B-3-1');
 OutTextXY(320,150,'B-2');
 OutTextXY(150,150,'B-4');
 OutTextXY(550,175,'48');
 OutTextXY(504,188,'60');
 OutTextXY(448,196,'72');
 OutTextXY(370,204,Chr(57)+Chr(50)+Chr(171));
 OutTextXY(270,205,'110');
 OutTextXY(193,204,'130');
 OutTextXY(125,203,'148');
 OutTextXY(91,197,'157');
 OutTextXY(40,192,'170');
 OutTextXY(5,180,'182');
 SetfillStyle(2,0);
 FillPoly(SizeOf(A34F) div SizeOf(PointType),A34F);
 FillPoly(SizeOf(C16F) div SizeOf(PointType),C16F);
 Line(479,137,479,166);
 Line(114,134,114,166);
 SetTextStyle(1,0,3);
 SetTextJustify(CenterText,CenterText);
 OutTextXY(320 , 250,'TCG M.F.CAKMAK  D351');
 OutTextXY(300,280,'CONFIGURATIONAL  PLAN  ,  FUEL  OIL  SERVICE
TANKS');
 Readln;
 end;
 CloseGraph;
 end;
 end.
```

```
unit dmw3;
[* this unit is called by unit dm8 and draws the feed and potable
  water tanks of the ship in the graphics mode *]
interface
uses Graph,Dos,TestGr;
const
Potable_Rectangle : array[ 1..5 ] of PointType = ((x:20; y:40),
                        (x:60; y:40),
                        (x:60; y:60),
                        (x:20; y:60),
                        (x:20; y:40));
B7W : array[ 1..5 ] of PointType = ((x:412; y:181),
                        (x:453; y:178),
                                (x:453; y:191),
                                (x:412; y:194),
                                (x:412; y:181));
B8W : array[ 1..5 ] of PointType = ((x:412; y:106),
                        (x:453; y:109),
                                (x:453; y:122),
                                (x:412; y:119),
                                (x:412; y:106));
B11W : array[ 1..5 ] of PointType = ((x:236; y:185),
                         (x:275; y:185),
                                  (x:275; y:200),
                                  (x:236; y:200),
                                  (x:236; y:185));

B12W : array[ 1..5 ] of PointType = ((x:236; y:100),
                         (x:275; y:100),
                                  (x:275; y:115),
                                  (x:236; y:115),
                                  (x:236; y:100));

LoadingShip : array [ 1..46 ] of PointType = (( x:10 ; y:125),
                                ( x:45 ; y:113),
```

(x:96 ; y:108),
(x:114; y:105),
(x:130; y:104),
(x:130; y:102),
(x:198; y:101),
(x:236; y:100),
(x:275; y:100),
(x:278; y:100),
(x:306; y:101),
(x:374; y:103),
(x:412; y:106),
(x:453; y:109),
(x:453; y:111),
(x:456; y:113),
(x:475; y:113),
(x:509; y:116),
(x:509; y:125),
(x:554; y:129),
(x:682; y:145),
(x:690; y:147),
(x:695; y:150),
(x:690; y:153),
(x:682; y:155),
(x:554; y:171),
(x:509; y:175),
(x:509; y:184),
(x:475; y:187),
(x:456; y:187),
(x:453; y:189),
(x:453; y:191),
(x:412; y:194),
(x:374; y:197),
(x:306; y:199),
(x:278; y:200),
(x:275; y:200),

```
                                        ( x:236; y:200),
                                        ( x:198; y:199),
                                        ( x:130; y:198),
                                        ( x:130; y:196),
                                        ( x:114; y:195),
                                        ( x:96;  y:192),
                              ( x:45;  y:187),
                                        ( x:10;  y:175),
                                        ( x:10;  y:125));
procedure main;
implementation
procedure main;
begin
if not Test_Graph_Device(GraphDriver,ErrorCode,GraphMode) then
 begin
  writeln('Program aborted');
 Halt(1);
 end
else
begin
 DrawPoly(SizeOf(LoadingShip) div SizeOf(PointType),LoadingShip);
 Line(10,134,130,134);
 Line(10,166,130,166);
 Line(45,113,45,187);
 Line(96,108,96,192);
 line(114,105,114,195);
 Line(130,102,130,198);
 Line(198,101,198,199);
 Line(236,100,236,115);
 Line(236,185,236,200);
 Line(275,100,275,200);
 Line(278,100,278,200);
 Line(306,101,306,199);
 Line(278,131,306,131);
 Line(278,168,306,168);
```

```
Line(374,103,374,197);
Line(412,106,412,119);
Line(412,181,412,194);
Line(453,109,453,191);
Line(456,111,456,189);
Line(479,113,479,187);
Line(509,116,509,184);
Line(559,129,559,171);
line(678,145,678,155);
Line(690,147,690,153);
Line(695,148,695,152);
Line(198,115,275,115);
Line(198,185,275,185);
Line(374,116,453,122);
Line(374,184,453,178);
Line(456,137,509,137);
Line(456,166,509,166);
Line(509,159,524,159);
Line(524,159,524,150);
Line(524,150,559,150);
OutTextXY(390,150,'B-1-1');
OutTextXY(220,150,'B-3-1');
OutTextXY(320,150,'B-2');
OutTextXY(150,150,'B-4');
OutTextXY(550,175,'48');
OutTextXY(504,188,'60');
OutTextXY(448,196,'72');
OutTextXY(370,204,Chr(57) + Chr(50) + Chr(171));
OutTextXY(270,205,'110');
OutTextXY(193,204,'130');
OutTextXY(125,203,'148');
OutTextXY(91,197,'157');
OutTextXY(40,192,'170');
OutTextXY(5,180,'182');
Rectangle(20,10,60,30);
```

```
FloodFill(25,25,14);
OutTextXY(70,20,'FEED WATER');
Floodfill(210,112,14);
Floodfill(210,190,14);
Floodfill(380,110,14);
Floodfill(380,190,14);
SetfillStyle(10,0);
FillPoly(SizeOf(B8W) div SizeOf(PointType),B8W);
FillPoly(SizeOf(B7W) div SizeOf(PointType),B7W);
FillPoly(SizeOf(B11W) div SizeOf(PointType),B11W);
FillPoly(SizeOf(B12W) div SizeOf(PointType),B12W);
FillPoly(SizeOf(Potable_Rectangle) div SizeOf(PointType),
                                                Potable_Rectangle);
OutTextXY(70,50,'POTABLE WATER');
OutTextXY(435,95,'B-8W');
OutTextXY(375,90,'B-10W');
OutTextXY(245,90,'B-12W');
OutTextXY(190,90,'B-14W');
SetTextStyle(1,0,3);
SetTextJustify(CenterText,CenterText);
OutTextXY(320 , 250,'TCG M.F.CAKMAK  D351');
OutTextXY(300,280,'CONFIGURATIONAL PLAN , WATER TANKS');
Readln;
end;
CloseGraph;
end;
end.
```

fire_stations data file

[* This data file is used by unit dm6.tpu *]

```
5  74   2  B-1-1 FWD FIREROOM
5  81   0  B-1-1 FWD FIREROOM
5  82   0  B-1-1 FWD FIREROOM
5  94   0  B-2   FWD ENGINEROOM
5  103  0  B-2   FWD ENGINEROOM
5  108  0  B-2   FWD ENGINEROOM
5  114  0  B-3-1 AFT FIREROOM
5  117  1  B-3-1 AFT FIREROOM
5  117  3  B-3-1 AFT FIREROOM
5  140  1  B-4   AFT ENGINEROOM
5  140  3  B-4   AFT ENGINEROOM
5  141  0  B-4   AFT ENGINEROOM
5  142  0  B-4   AFT ENGINEROOM
3  48   1  A-305L CREW'S QUARTERS
2  18   0  A-203LM CPO MESSROOM
2  40   0  A-204-2L PASSAGE
2  69   2  A-208L FOOD SERVICE
2  71   0  A-207L PASSAGE
2  169  0  C-203L CREW QUARTERS
2  170  0  C-204LM CREW QUARTERS
2  195  0  C-205L CREW QUARTERS
1  19   0    WEATHER
1  41   0    WEATHER
1  41   2    WEATHER
1  41   4    WEATHER
1  41   6    WEATHER
1  41   8    WEATHER
1  69   0    WEATHER
1  80   0    WEATHER
1  100  0    WEATHER
1  118  0    WEATHER
1  141  1    WEATHER
1  141  2    WEATHER
```

1 147 0 B-104ACEL PASSAGE
1 171 1 WEATHER
1 171 2 WEATHER
1 171 4 WEATHER

V. CONCLUSIONS AND RECOMMENDATIONS

This Chapter presents the conclusions, recommendations and the future software implementations.

A. CONCLUSIONS

A local area network will greatly increase the efficiency and functionality of Turkish Battleships. A local area network is a cost-effective multiuser system. The personal computers in the local area network do not occupy much space in the battleship; neither do they require special air conditioning. The programs, data and resources can be shared among the users. This feature of the local area network also reduces the marginal cost of the software and the peripheral devices.

Although the computers in the local area network can be based on any Intel family microprocessors, having Intel 80386-type microprocessors will increase efficiency and accuracy while decreasing response time. The price difference between the 80286 and the 80386 is not great. This fact has prompted us to employ Intel 80386 microprocessors.

This thesis has mainly concentrated on creating an application software library for Turkish Battleships. The three application software package has already been implemented in this thesis. This package will meet some of the most important software requirements of Turkish Battleships. The effort for creating the military oriented software packages will continue.

The Turbo Pascal 4.0 was employed in the implementation of the software package. We were planning to employ Ada in the beginning, but the available Ada compilers did not support graphic capabilities at all. This consideration led us to employ Turbo Pascal 4.0. The software package created for Turkish Battleships can also run on stand-alone computers. This allows users to run the programs even when they do not have multiuser network environment on a small battleship. This feature also permits programs to be run in educational centers for training purposes. The major hardware requirement of the programs is an IBM or equivalent based computer (or computers in a multiuser network environment). We have implemented both monochrome and EGA monitor versions of the package. In order to succesfully run the programs, the computers should have the following files:

1. Dos.doc,

2. Crt.doc,
3. Graph.doc,
4. Fire.txt,
5. MFCakmak.exe,
6. Trip.chr,
7. Hercules.bgi for monochrome monitors or Egavga.bgi for EGA monitors.

Although almost every network software provides password operated sign on, in order to assure security we also provided a second step password in our implementation. The LAN is also one of the best solutions to the hardware security problems. As long as the server does not have a hardware problem, the entire system will not be affected by the users hardware problems.

B. RECOMMENDATIONS AND FUTURE IMPLEMENTATIONS

The software requirements of the navy and the battleships are quite numerous. Turkish Navy should continue to implement other software requirements. The Turkish Navy should also organize a Software Development, Research and Coordination Branch under the command of the Technical Department. This branch should plan, organize, develop, test and control the future software implementations. The Turkish Navy already has the required man-power for this Software Development, Research and Coordination Branch. The following features should be implemented and combined in TCG. MF. Cakmak in the near future:

1. Mouse capability in CIC and tank diagrams,
2. Getting CIC target data and draft values from the electromechanic sensors,
3. Supply Office COSAL manager program should be implemented,
4. Electrical Configuration of the ship should be implemented,
5. Trouble shouting can easily be implemented,
6. Turkish character set and word processing should be combined [Ref. 6],
7. Tank sound control should be implemented,
8. Computer aided navigation should be advanced,
9. The computer aided missile path design should be implemented.

The users of programs created for Turkish Battleships will not be computer scientists, nor programmers. The Turkish Navy should also organize short term courses to teach computer operators how to run the computer programs.

REFERENCES

1. Berry, Paul, *Operating the IBM PC Networks,* Sybex, San Francisco, CA, 1985.
2. Stallings, William, *Data and Computer Communications, 2nd. edition,* Macmillan Publishing Co. NY, 1988.
3. Deitel, Harvey, *An Introduction to Operating Systems,* Boston College, MA, July 1984.
4. Intel Corporation, *80386 Reference Manual,* Intel, Santa Clara, CA, 1987.
5. Berzins, Valdis, *Software Engineering,* class notes for CS 4500 at the Naval Postgraduate School, Monterey, CA, October 1988.
6. Akinci, Metin, *Turkish Character Set Generator Implementation,* Master's Thesis, Naval Postgraduate School, Monterey, CA, June 1988.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Professor Uno Kodres, Code 52Kr Department of Computer Science Naval Postgraduate School Monterey, CA 93943-5000	2
5.	Tarik Gokasan Yapi ve Kredi Bankasi Balat Subesi Istanbul / TURKEY	1
6.	Nevzat Gulesen Yavuz Selim Cad. Silistre Sok. 27/3 Fatih - Istanbul / TURKEY	2
7.	Genel Kurmay Baskanligi Personel Dairesi ve AR-GE Baskanligi Bakanliklar - Ankara / TURKEY	2
8.	Kara Kuvvetleri Komutanligi Personel Daire Baskanligi, Topculuk Subesi ve Silah Sistemleri Muhendisligi Subesi Bakanliklar - Ankara / TURKEY	3
9.	Topcu ve Fuze Okulu Ogretim Kurulu AR-GE Subesi Polatli - Ankara / TURKEY	1
10.	Kara Harp Okulu Komutanligi Okul Kutuphanesi ve Elektrik Bolumu Kutuphanesi Bakanliklar - Ankara / TURKEY	2

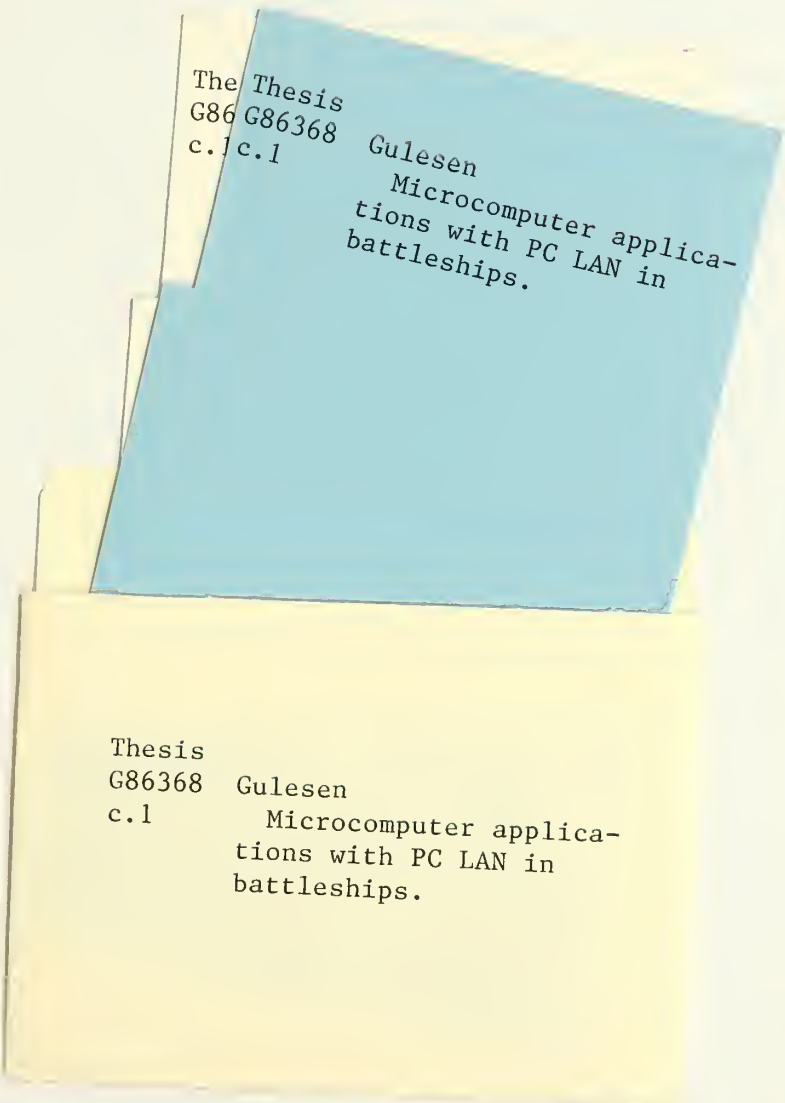
11. Deniz Harp Okulu Komutanligi
Okul Kutuphanesi ve
Elektrik Bolumu Kutuphanesi
Tuzla - Istanbul / TURKEY — 2

12. Fakulte ve Yuksek Okullar Komutanligi
Kutuphanesi
Dikimevi - Ankara / TURKEY — 1

13. Fakulte ve Yuksek Okullar Komutanligi
Kutuphanesi
Cankurtaran - Istanbul / TURKEY — 1

14. Bogazici Universitesi
Elektrik Fakultesi
Istanbul / TURKEY — 1

15. Ortadogu Teknik Universitesi
Elektrik Fakultesi
Ankara / TURKEY — 1



Thesis
G86368
c.1
Gulesen
Microcomputer applica-
tions with PC LAN in
battleships.